Deploying Oracle ADF JClient Applications with Java Web Start

An Oracle WhitePaper
May 2004

ORACLE

Deploying Oracle ADF JClient Applications with Java Web Start

Deploying Oracle ADF JClient Applications with Java Web Start

INTRODUCTION

This document is a detailed guide on how to deploy ADF JClient applications, built on top of ADF Business Components, with Java Web Start. Reading this whitepaper you learn how to deploy and run ADF JClient applications with Java Web Start using

- The embedded OC4J server in Oracle JDeveloper 10*g*
- A standalone OC4J server
- The Oracle Application Server 10*g*
- An ADF Business Components model deployed as EJB
- JBoss

It is assumed that the reader knows about how to build ADF JClient applications based on ADF Business Components, and that Java Web Start is installed on the target client machines.

The Java Web Start software can be downloaded for free from the Java SUN website[1].

ABOUT ADF JCLIENT

For an overview of ADF JClient, please read "Developing Swing-based Java Clients using Oracle JDeveloper 10*g* and Oracle ADF JClient" available at http://otn.oracle.com/products/jdev

Oracle JDeveloper 10*g* contains Oracle ADF JClient, the Oracle development and deployment environment for databound Java Swing applications.

ADF JClient leverages the new Oracle Application Development Framework (Oracle ADF) to declaratively bind Swing components to business services like ADF Business Components, Enterprise JavaBeans, Webservices, TopLink and JavaBeans.

Using the SUN Java Web Start technology, Oracle ADF JClient applications can be deployed local to a client while still enabling a centralized software management.

[1] http:/ / java.sun.com/ products/ javawebstart/ index.jsp or following the "Free Download" link on Sun's "consumer friendly Web site", http:/ / java.com

JAVA WEB START

For a detailed description of what Java Web Start is and for further documentation, please visit the Java Web Start product page at SUN

http://java.sun.com/products/javawebstart/reference/index.html

Java Web Start is part of the SUN J2SE 1.4 platform and provides a standard deployment solution for Java Swing based applications[2].

Invented by SUN Microsystems Inc, Java Web Start provides a platform-independent, secure, and robust deployment technology enabling developers to deploy full-featured applications to end-users by making the applications available on a standard web server.

The application deployer, through configuration, decides whether the end-user is offered the ability to create a desktop shortcut for the downloaded application. When using a desktop shortcut, applications run on Java Web Start look like any local installed desktop application.

How Java Web Start works

Java Web Start is based on Java Network Launching Protocol (JNLP) technology.

JNLP is a web-centric protocol for provisioning and running J2SE based applications.

The specification of JNLP[3] includes:

- A web-centric application model with no installation phase, providing transparent and incremental updates.
- The syntax and semantics of a JNLP file, which packages applications on a web server and delivers and runs them on clients.
- A standard execution environment for an application, which is similar to the sandbox model but enhanced with additional APIs (e.g. for local clipboard and disk access). Standard Java code signing techniques can be used to grant additional permissions to the executed code in Web Start.

Why Java Web Start?

Java Web Start provides the benefits of client-side software distribution, while still allowing a centralized software management. It's an easy, robust, and secure way to deploy applications to the Internet and intranet.

Some of the ease-of-use and ease-of-programming features associated with Java Web Start include:

Portability: Java Web Start is available on any platform be it Windows, Solaris, or Linux. Other client platforms are to follow.

---

[2] Java Web Start is also available for the Java platform priot to 1.4 but needs to be downloaded and installed separately

[3] http:/ / java.sun.com/ products/ javawebstart/ download-spec.html

Caching: Applications launched with Java Web Start are cached locally on the client. A downloaded Java Web Start application launches on par with a traditionally installed application.

Maintainability: An update of the remote application files leads to Java Web Start updating the locally cached version of the same application by its next invocation.

Easy launching: Java Web Start allows applications to be launched independently of a web browser. Optionally, desktop shortcuts are created, making Java Web Start application look like native client applications.

Ability to work offline: An application can be used in situations where launching through the browser is inconvenient or impossible.

EXAMPLE APPLICATION USED IN THIS PAPER

As it always seems easier to explain technical content by example, this paper uses a simple ADF JClient form, shown in the image below. The example application is based on ADF Business Components and is created out of a JFrame showing a master-detail relationship and a Business Intelligence Bean graph component.

The ADF Business Components application module is named "AppModule" and will be referenced later in this document when discussing the deployment to Enterprise JavaBeans.

The Data Control name used is "AppModuleDataControl" and can be accessed in the Structure window after selecting the DataBindings.cpx file in the Application Navigator.

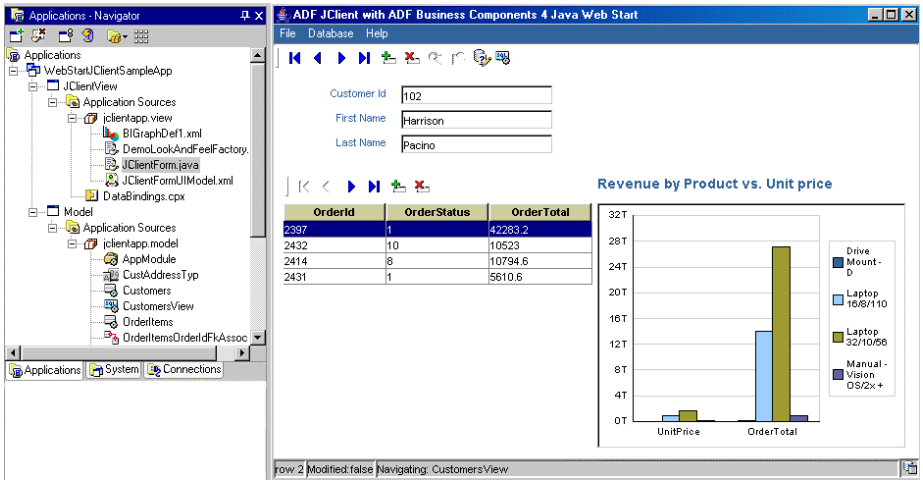

Figure 1: Example application used to explain the ADF JClient deployment with Java Web Start

The screenshot above shows the application in the JDeveloper 10*g* Application Navigator (Ctrl+Shift+A) and the Java Visual Editor. The Application Navigator is a structured view to the application source and meta-data files, showing only those files required for editing and deploying an application.

PREREQUISITES

This section describes how to create a enterprise deployment archive (EAR) for the ADF Business Components library files and how to create a self-signed certificate to establish trust to the user. This action is only required once after installing Oracle JDeveloper.

Trust no one

By default, Java Web Start applications execute within a security sandbox with limited access to the client desktop. The security sandbox is part of the J2EE security platform and prevents downloaded code from performing sensitive client-side operations without the end-user first granting permission.

Java applications that require desktop access therefore must authenticate itself to the user in a way that makes him – the end user – trust the downloaded code. A relationship of trust is established by code signing the Java deployment files with a key and certificate that identifies the application deployer to the end user.

To prove on the Internet that a certificate identifies a trustworthy entity, Trust Centers like Verisign or Thwate exist. Because certificates need to be purchased from Trust Centers, companies may decide to be their own Trust Center for applications deployed to the Intranet, using software like Oracle Certificate Authority in Oracle Application Server 10*g*.

For testing purposes, however, a self-signed certificate, that is a certificate issued by the same person requesting it, is sufficient.

In order for ADF JClient to work in a Java Web Start environment, requires full desktop access, which is why the process of code signing is explained more in detail in the following.

Creating a self signed certificate

To manage certificates, the Java SDK provides an administrative utility – "keytool" – that can be used to create certificates and sign code sources. If not specified different, then, on Windows, the keystore file that contains all user keys is located in the `\Documents and Settings\<username>` directory. If this file doesn't exist, it is created by the keytool.

Private keys and their associated public-key certificates are stored in password-protected databases called *keystores*. The filename by convention is "*.keystore*" and is located in the user home directory

For the examples in this whitepaper, we create a certificate with the name of "ADFJClientTrust".

To create the key, navigate to the bin directory of a Java SDK installation, like the `JDK\bin` directory of each Oracle JDeveloper installation and issue the following command on a command line:

Alias: A shortened name for an entity that has a key or certificate in the keystore.

```
Keytool –genkey  -keyalg rsa  -alias ADFJClientTrust
```

Each key entry can be protected by its own password, which in a production environment shouldn't be the same as the password chosen for the keystore itself.

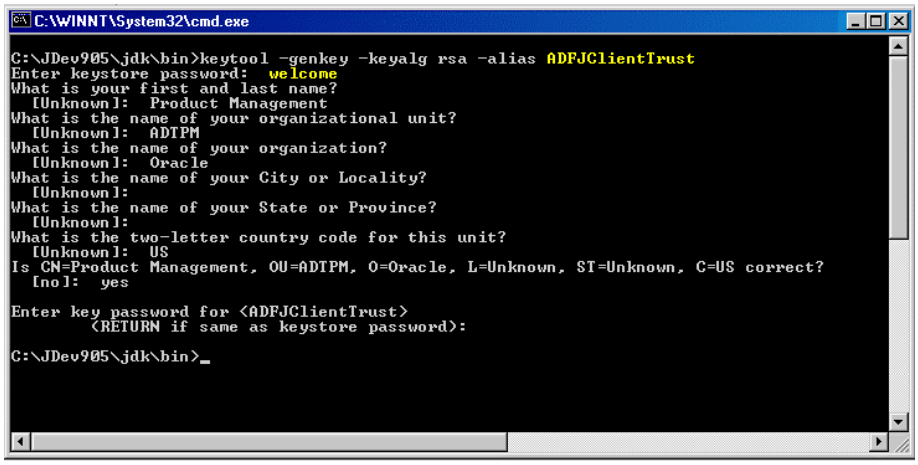

Figure 2: Keytool usage for creating a private key

This command creates a key named "ADFJClientTrust" and stores it in the `.keystore` file. The key and the password are used in the script to sign the `bc4jlibs.ear` file and later in JDeveloper when signing the Java Web Start client deployment files.

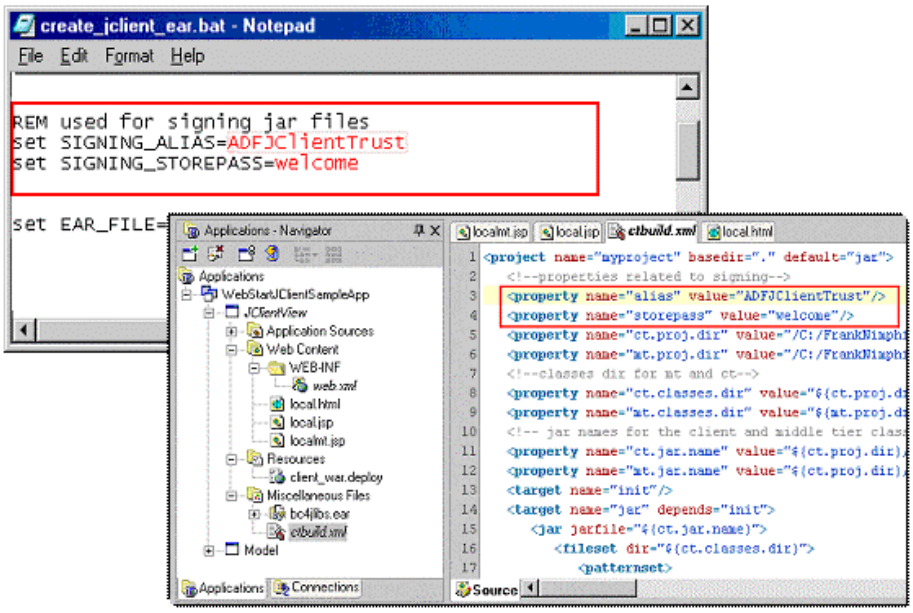

Figure 3: Using the credentials in the create_jclient_ear.bat file and in JDeveloper

The keystore password, "welcome" in this example, is to protect all keys in the keystore and is required later to sign code sources. A key password protects each individual key in a keystore and, if not specified, has the same password as the keystore.

☑ **Note**: The signing scripts provided by Oracle JDeveloper 10g need modifications if the password used for the key is different than the password used for the keystore. The required changes are explained later in this document.

Other information, like the organization and the state, are optional and define the distinguished name (DN) of the key, which is used by certificate authorities to uniquely identify the owner of a certificate. The DN is not interesting when working with self-signed certificates such those used in this paper.

The validity of a self-signed certificate is 90 days by default. You can extend validity by specifying the `–validity nDays` option when creating the key:

```
Keytool –genkey  -keyalg rsa  -alias ADFJClientTrust
–validity 360
```

More information about using certificates in Java is provided in Chapter 6, "Security Management" of the SUN documentation "Java™ 2 Platform Security Architecture"[4].

Creating and signing the ADF Business Components EAR file

The deployment file `bc4jlibs.ear` for the ADF Business Components libraries need only to be created once after installing Oracle JDeveloper 10*g*. The step of creating the EAR file is a necessary prerequisite for running the Java Web Start wizard in Oracle JDeveloper 10*g*. The Java Web Start wizard will not run if the `bc4jlibs.ear` file isn't present in the *<Jdeveloper_Home>*`\Bc4j\jlibs` directory.

To create the EAR file, a script, `create_jclient_ear.bat` for Windows and `create_jclient_ear.sh` for Unix, is provided in the *<JDeveloper Home>*`\Bc4j\bin` directory. The same script is used to sign the library file with the key created before.

Open the `create_jclient_ear.bat` or `create_jclient_ear.sh` file in a text editor to provide the alias name and the password specified when creating the self-signed certificate.

For the self-signed certificate created earlier in this paper, the alias name is "ADFJClientTrust" and the password "welcome".

To execute the script, open a command line and issue the following commands

1. `set Oracle_Home=`*`<JDeveloper_Home>`*

   `<JDeveloper_Home>` represents the name of the directory of the JDeveloper installation.

   For example: set `Oracle_Home=C:\Jdeveloper10g`

2. `create_jclient_ear.bat sign`

[4] http:/ / java.sun.com/ j2se/ 1.4.2/ docs/ guide/ security/ spec/ security-spec.doc.html

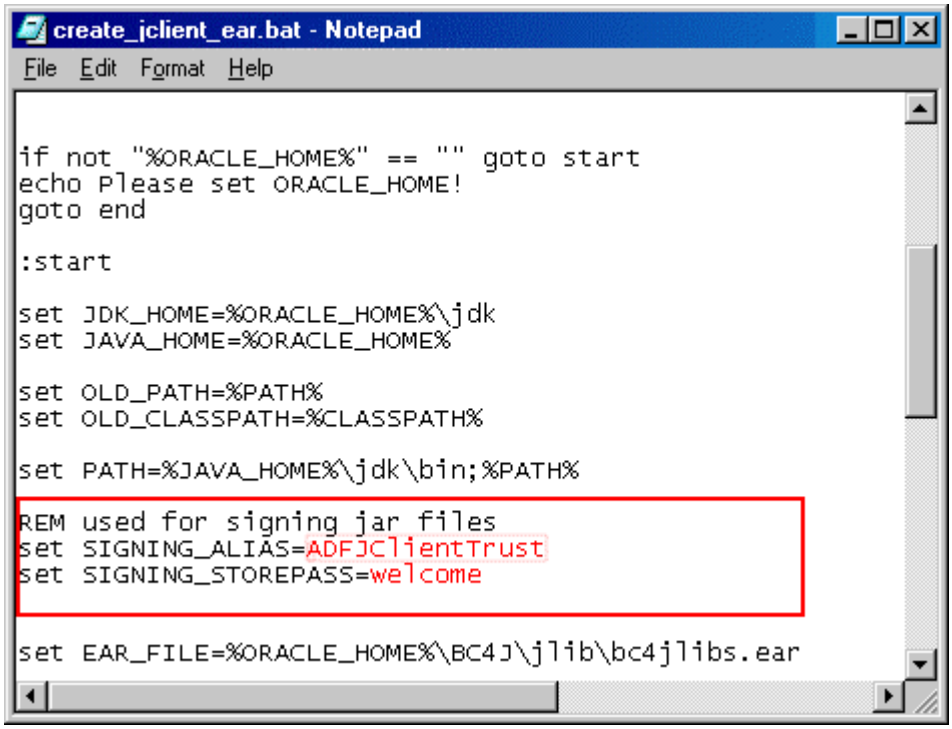

Figure 4: create_jclient_ear.bat

Make sure that at the end of the console output, it is reported that the script ran successfully. The "sign" argument in the command line makes sure the EAR file gets signed.

The generated output file `bc4jlibs.ear` is located in the `<JDeveloper_Home>\Bc4j\jlibs` directory.

☑ **Note**: If your key password is not the same as the keystore password, the following changes need to be done in the script

```
…
set SIGNING_ALIAS=<Your Signing Alias>
set SIGNING_STOREPASS=<Your Keystore Password>
set SIGNING_KEYPASS=<Your Key Password>
…

rem all in one line
set PROP_ARGS=-Djdeveloper.root=%ORACLE_HOME% -
Dear.file.name=%EAR_FILE% -Dalias=%SIGNING_ALIAS% -
Dstorepass=%SIGNING_STOREPASS% -keypass=%SIGNING_KEYPASS%
…
```

LOCAL DEPLOYMENT WITH JAVA WEB START USING OC4J

For a JClient application to be deployed with Java Web Start, a JNLP file needs to be created. To create a JNLP file, open the ADF JClient project in JDeveloper and select **New** from the right mouse context menu

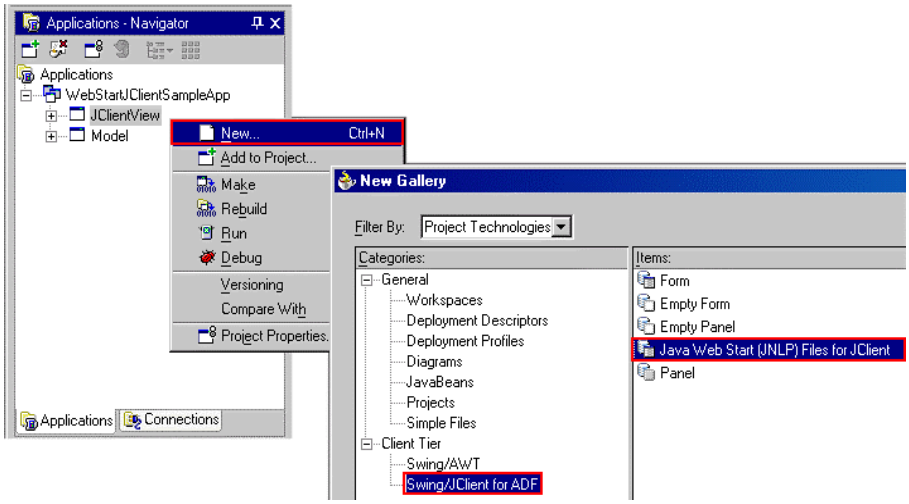

Figure 5: Launching the JNLP file wizard for ADF JClient

The Java Web Start JNLP file is created for only one Data Control file, which is selected in step 1 of the wizard.

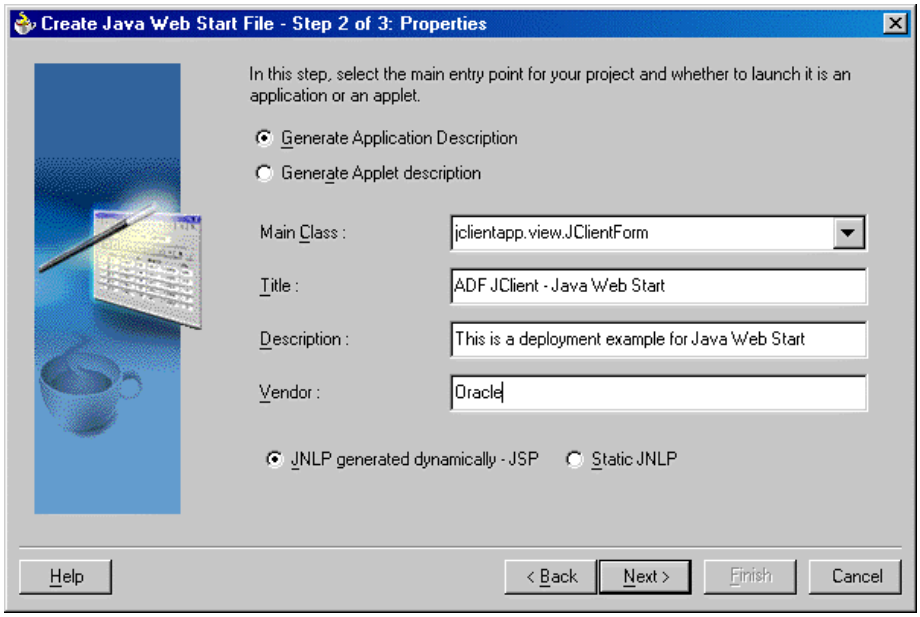

Figure 6: Properties dialog for the Java Web Start deployment

The second wizard panel defines the JClient application's main class that is called immediately when launching Java Web Start. All other information is displayed to the user in the Java Web Start splash screen.

The Java Web Start File wizard provides options to dynamically create a JNLP file, by using a JSP file on the server to generate the JNLP file upon a user request, and to create a static JNLP file that contains all start parameters required to launch Java Web Start.
Dynamically creating a JNLP file is the recommended option and also the default. In the following this option is further explored.

In the third wizard step, optionally you can add the graph library files required when using the Business Intelligence Graph Bean for rendering charts in ADF JClient.

Beside of the Web content files `local.jsp` and `localmt.jsp`, if the dynamic JNLP file generation option was selected in the JClient Java Web Start wizard, an ANT build script `ctbuild.xml` is created to package the ADF JClient application for the Java Web Start deployment.

☑ **Note**: You can rename the two JSP files if changing the name references in `local.jsp` for `localmt.jsp`.

Because the ADF JClient application files must be signed, the certificate alias and the keystore password must be provided in the `ctbuild.ctx` file. To edit this file in Oracle JDeveloper, double-click its node in the Application Navigator.

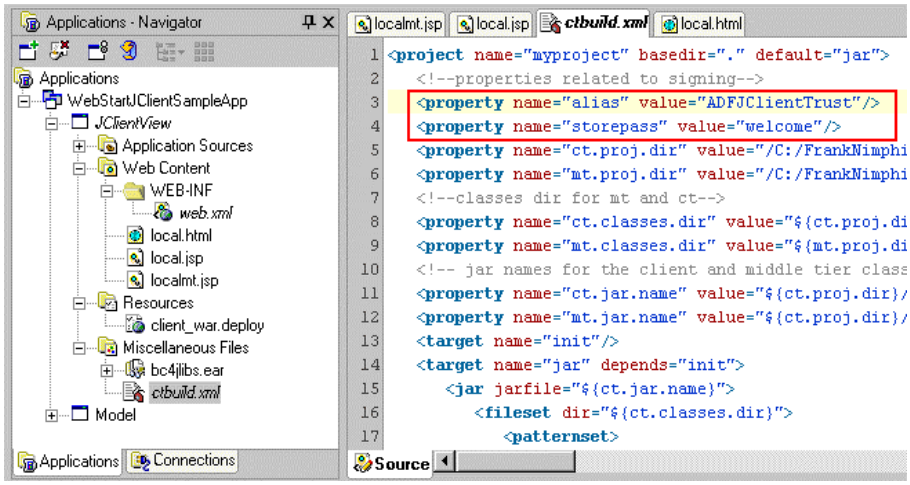

Figure 7: Providing signing alias and store password in the ANT build script

☑ **Note**: If your key password is not the same as the keystore password, the following changes need to be done in the script

```
…
<property name="storepass" value="<store password"/>
<property name="keypass" value="<key password"/>
…

<target name="sign" depends="jar">
   <signjar jar="${mt.jar.name}" alias="${alias}"
   keypass="${keypass}" storepass="${storepass}"/>
   <signjar jar="${ct.jar.name}" alias="${alias}"
   keypass="${keypass}" storepass="${storepass}"/>
</target>
…
```

The self-signed certificate shown as an example in this whitepaper does not use a password different from the one specified for the keystore and therefore `ctbuild.xml` does not require modification.

As a last configuration step, before running ADF JClient with Java Web Start, you need to build and sign the ADF JClient application sources by selecting **Build Target → sign** from the right-mouse context menu as shown in the image below.

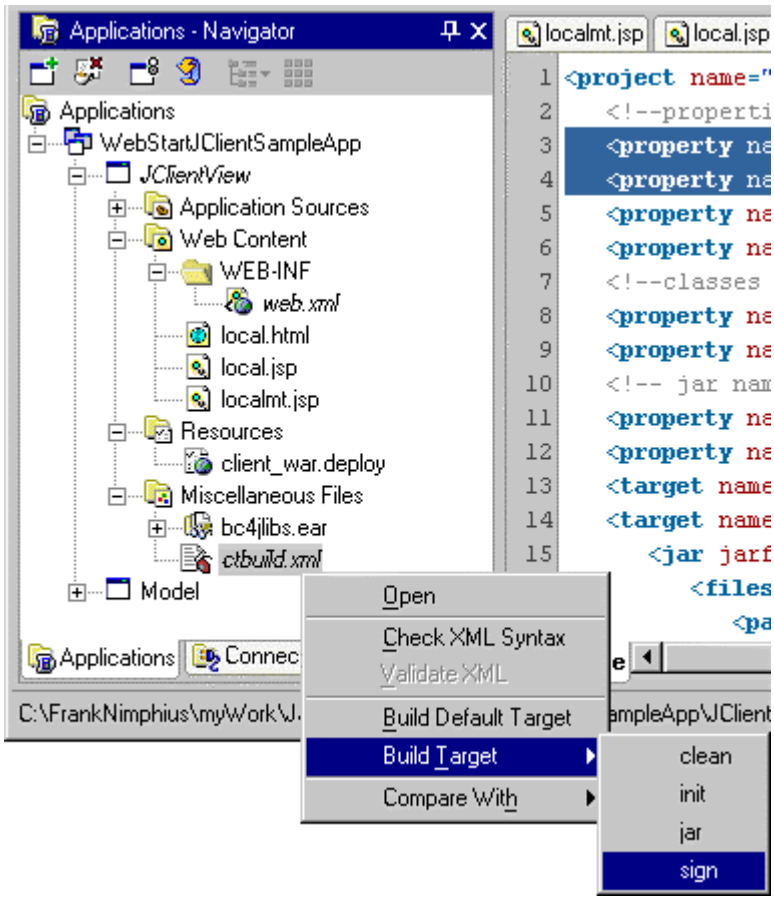

Figure 8: Creating and signing ADF JClient sources for running with Java Web Start

At the end of the build process, the Oracle JDeveloper log window should show a message similar to the following

```
BUILD SUCCESSFUL
Total time: 4 seconds
[3:40:24 PM] Successful compilation: 0 errors, 0 warnings.
```

Embedded OC4J

The embedded Oracle Containers for J2EE (OC4J) is a complete J2EE container within Oracle JDeveloper 10*g* that can be used for testing.

To test ADF JClient on Java Web Start, select the `local.html` file entry in the Application Navigator select **Run** from the right-mouse button menu or from the toplevel toolbar, or press the **F11** key. If OC4J wasn't started before, then Oracle JDeveloper starts it first.

In the opened web browser, select the "Launch JClient Project" link to start your ADF JClient application[5].

☑ **Note**: You can also directly run the `local.jsp` file.

The first time that an ADF JClient application is launched using Java Web Start, a security message will appear because the ADF JClient application requests all permissions.

Information, like the signer's common name, provided when signing the certificate will show in the message dialog for the user to decide whether or not to trust the code signer.

After the first initial download, the source files are cached on the client and will be downloaded in subsequence usages only if the application source files have changed on the server.

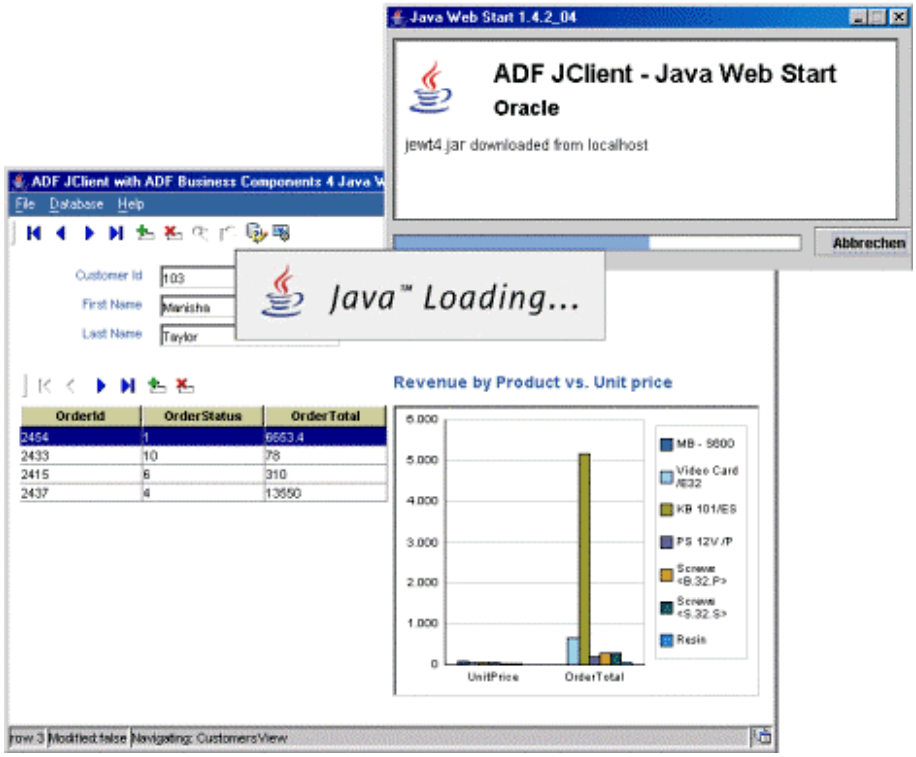

Figure 9: ADF JClient application running on Java Web Start

Java Web Start can be configured to create an application shortcut on the client desktop so that the application can be started like a client-server application, not requiring a browser. Please see the Java Web Start documentation for options on how to configure the JNLP file[6].

---

[5] If you are experiencing problems with this link and you are running Internet Explorer, please refer to the troubleshooting section at the end of this whitepaper

[6] http:/ / java.sun.com/ products/ javawebstart/ 1.2/ docs/ developersguide.html

Standalone OC4J

Running an ADF JClient applications with Java Web Start deployed on a stand-alone[7] OC4J server is similar to using the embedded OC4J instance. The difference between the standalone and the embedded OC4J version is that the ADF JClient application source first needs to be deployed to the OC4J server before the application can be requested on the Web.

Before deploying the ADF JClient application to the standalone OC4J server, you need to start up OC4J. For this, open a command line window and navigate to the `j2ee\home` directory of the OC4J installation[8]. Start OC4J by issuing the following command:

```
java –jar oc4j.jar
```

This starts OC4J with its predefined administrator password "welcome". To install OC4J so that the password can be changed and all text files get adjusted to the Operation System used, call:

```
java -jar oc4j.jar –install
```

After this, start OC4J as shown above. For this whitepaper the default admin password "'welcome" is used.

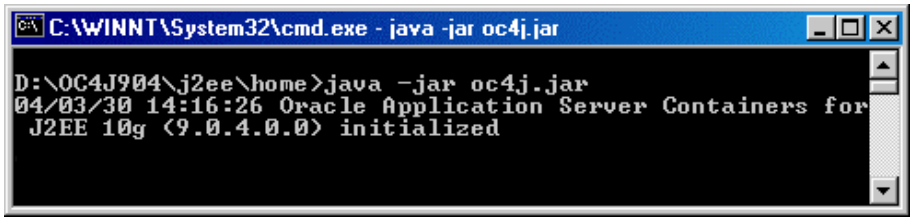

Figure 10: Starting stand-alone installation of Oracle Containers for J2EE

You can test the OC4J server instance by issuing `http://<server_name>:8888` in the URL filed of a browser[9].

Creating a Connection to OC4J

In Oracle JDeveloper 10*g*, open the "Connection Navigator" by selecting its entry in the IDE "View" menu, or, by pressing the Ctrl+Shift+O key combination.

The Connection Manager allows you to create predefined and named connections to application server instances, like OC4J, that can be used instead of typing the complete connect string when deploying applications.

---

[7] Stand-alone OC4J means that the Oracle J2EE container does not run in the context of an Oracle Application Server or Oracle JDeveloper installation. OC4J can be downloaded from http:/ / otn.oracle.com/ tech/ java/ oc4j/ index.html

[8] The OC4J installation process for the standalone J2EE container is to unzip the downloaded archive file.

[9] If this doesn't bring up the OC4J homepage, then check <OC4J Home>\ j2ee\ home\ config\ http-web-site.xml for a possible port change

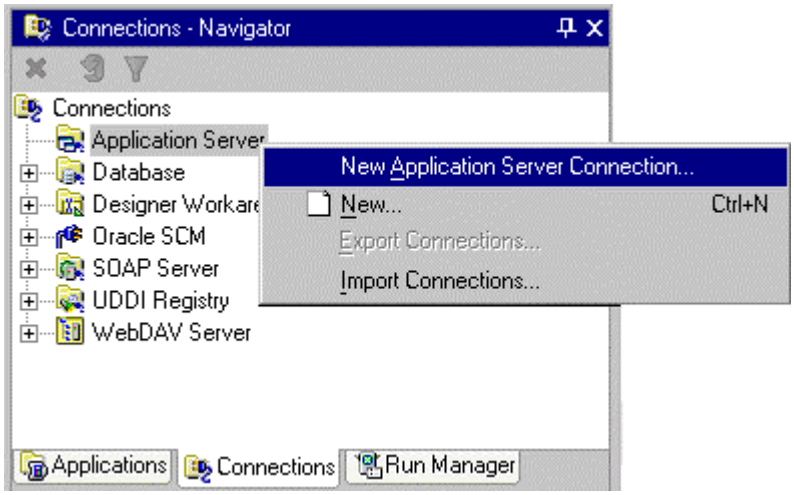

Figure 11: Connection Navigator in Oracle JDeveloper

For the example in this whitepaper, an OC4J connection with a name reference "OC4J_STAND_ALONE" gets created.

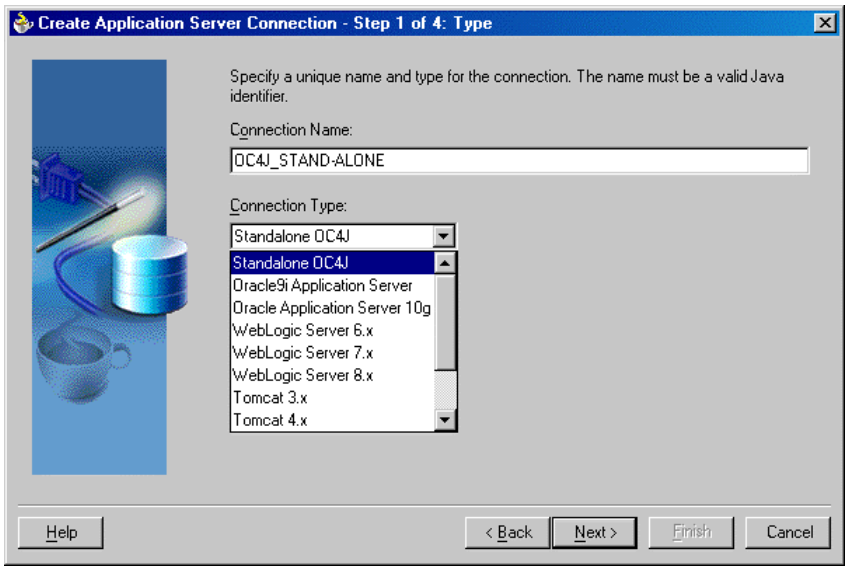

Figure 12: Application Server type selection

After selecting the type of Application Server that this named configuration should declaratively connect to, the second wizard panel expects the administrator name and password to be provided, which in a default OC4J installation is "admin" for the username and "welcome"[10] for the password.

The third wizard panel, the Connection panel, needs information about the server name and the physical location of the *<OC4J_Home>*`\j2ee\home`. The

[10] Though the password is encrypted, it is recommended to change the administrator password in the <OC4J Home>\ j2ee\ home\ config\ jazn-data.xml file. To change the admin password, install OC4J with `java -jar oc4j.jar –install`

<OC4J_Home>\j2ee\home directory hosts the admin.jar file, which is used by the Connection Manager to administer the OC4J instance.

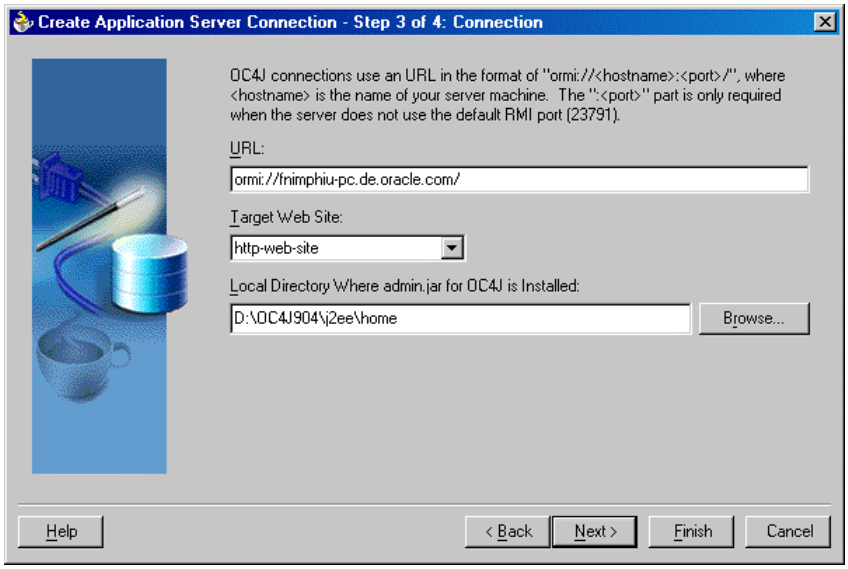

Figure 13: Provide the server name and the location of the OC4J admin.jar file

Deploying the bc4jlibs.ear file to stand-alone OC4J

The bc4jlibs.ear file can be deployed to OC4J by selecting its entry in the JClient project in the Application Navigator and choosing the standalone OC4J named connection created before.

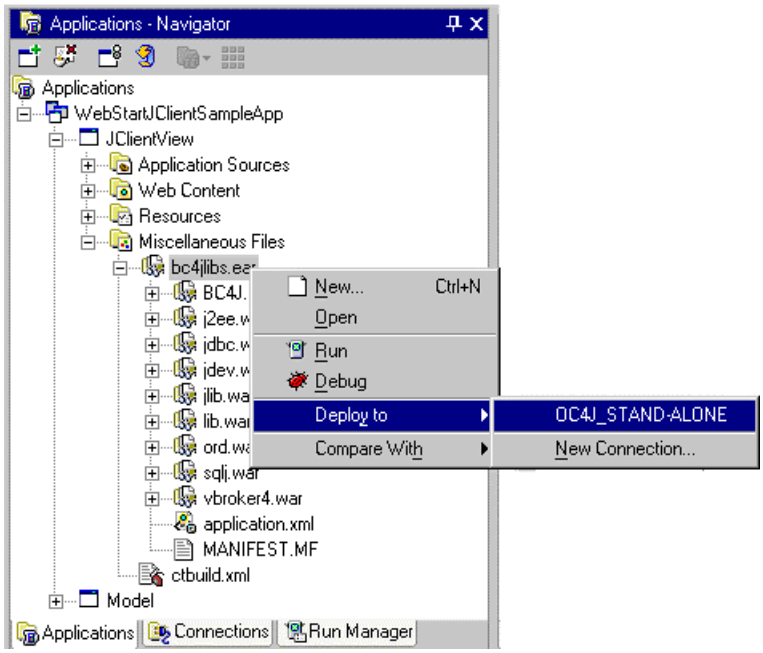

Figure 14: Deploying bc4jlibs.ear file to OC4J stand-alone

The `bc4jlibs.ear` file only needs to be deployed once, because its content does not change between applications provided that the code signature isn't changed.

Local Deployment to OC4J

When finishing the ADF JClient wizard for Java Web Start, a Web application deployment profile for the ADF JClient client project, `client_war.deploy`, automatically gets created. The deployment profile can be edited to customize the deployment configurations.

To edit the deployment profile, select the client_war.deploy entry under the Resource node. Use the right mouse button to open the context menu and choose the **Properties…** option. This opens the WAR Deployment Profile Properties dialog.

One property to customize is the "Web Application Context Root" name that by default is taken form the JDeveloper project settings. The context root name represents the virtual path in the application request URL, which is why a more comprehensive name than the default may be useful. For example, changing the Web application context root to "adfjclientToWebstart" makes the application request URL look like the following:

```
http://localhost:8888/adfjclientToWebstart/local.jsp
```

In addition to customizing the Web application root context name, the WAR Deployment Profile Properties dialog can also be used to include additional libraries to the deployment or to define dependency filters that only add those classes and their dependencies to the deployment profile that are really used in an application.

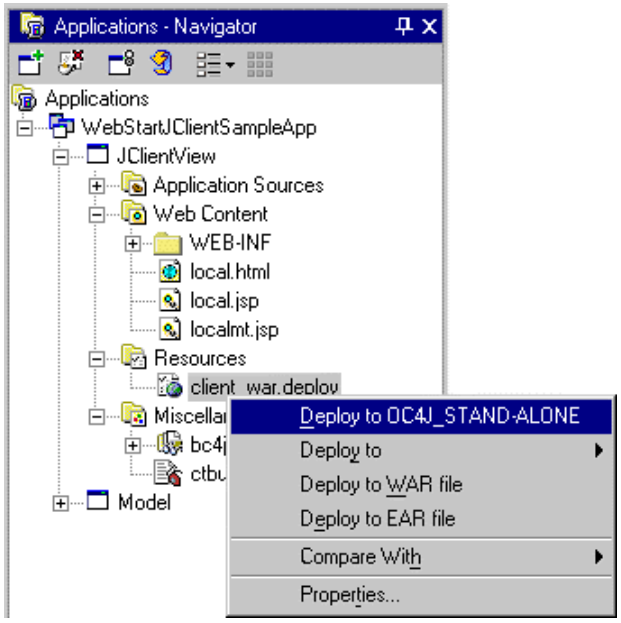

Figure 15: deploy the ADF JClient application to the stand-alone OC4J

To deploy the application to OC4J, select the "client_war.deploy" entry in the Application Navigator and choose the **Deploy to OC4J_STAND-ALONE** entry in the right-mouse context menu.

After the deployment has finished, the following lines are shown in the log message window of JDeveloper:

```
Use the following context root(s) to test your web
application(s):
http://<server name>:8888/adfjclientToWebstart
Elapsed time for deployment:  7 seconds----  Deployment
finished.
```

As before, when deploying ADF JClient to the embedded OC4J server instance, the application start file that dynamically creates the Java Web Start JNLP file is `local.jsp`. The application URL in this example therefore is:

```
http://<server name>:8888/adfjclientToWebstart/local.jsp
```

Local Deployment to Oracle Application Server 10*g*

This paper does not discuss Oracle Application Server 10g in detail. Please read http://otn.oracle.com/products/ias/htdocs/as10gfaq.html for general Oracle Application Server questions.

The deployment steps required to run the ADF JClient application from Oracle Application Server 10*g* are similar to the steps explained for standalone OC4J.

It is possible to deploy ADF JClient to Oracle Application Server directly from Oracle JDeveloper 10*g* or by using Oracle Application Server Control 10*g*, a web-base administration tool. Both deployments are explained in this paper. This section begins by using Oracle JDeveloper 10*g* to deploy to Oracle Application Server.

To avoid conflicting with other OC4J instances running on the Oracle Application Server 10*g* instance, Oracle Application Server Control 10*g* is used to create a new OC4J instance with the name "adfjclientToWebstart"[11] to experiment with ADF JClient deployments.

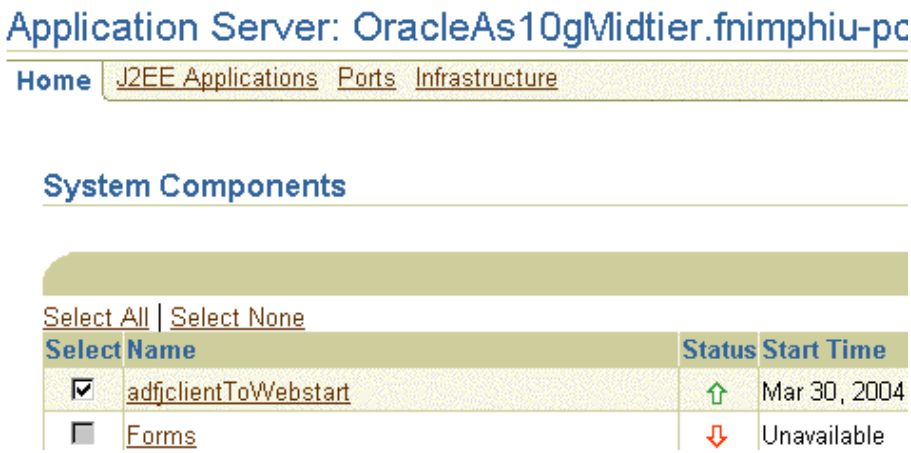

Figure 16: OC4J instance "adfjclientToWebstart" in Oracle Application Server Control 10*g*

Creating and starting a new OC4J instance in Oracle Application Server Control 10*g* is an intuitive task and doesn't require detailed description.

As the image above indicates, the OC4J instance for ADF JClient is started, which is a requirement for deploying the application.

Similar to the standalone OC4J deployment, the Connection Navigator in Oracle JDeveloper 10*g* is used to create a named connection to Oracle Application Server 10*g*. Open the Connection Navigator (Ctrl+Shift+O) and select the Application Server node. Use the right mouse button to open the context menu and select **New Application Server Connection …** to create a new entry for the Oracle Application Server.

[11] It is not a must to create a new OC4J instance for ADF JClient. However, having a separate OC4J instance for ADF JClient is more convenient when a restart of the OC4J instane is required.

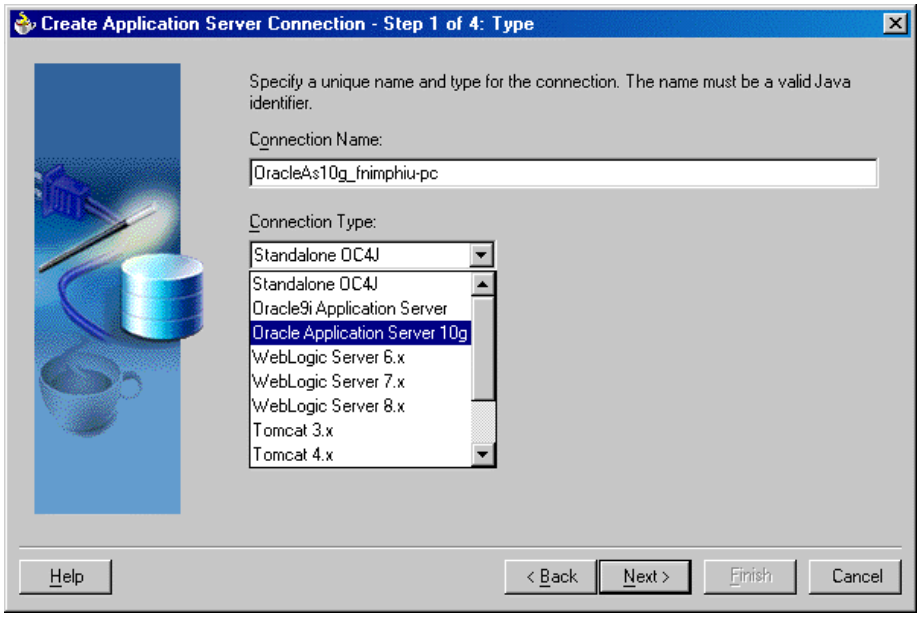

Figure 17: Define the Connection type as Oracle Application Server 10g

On the first wizard page, provide a name for the application server connection and chose the connection type as **Oracle Application Server 10*g***.

The second wizard page requires you to provide the account information for the ias_admin user.

Two ias_admin accounts are available after installing Oracle Application Server 10*g*, one for the Oracle Application Server infrastructure installation and one for the application server middle tier installation. ADF JClient applications, like other J2EE applications, are deployed to the application server middle tier and therefore the ias_admin account for this installation is required.

The third wizard page expects access path information for the Oracle Application Server Control 10*g* to be provided.

The default ports used for Oracle Application Server Control 10*g* are 1810 for the infrastructure installation and 1811 for the middle tier installation. Because Oracle Application Server automatically handles port conflicts during installation, it could be that the port numbers are different from the default. To lookup the port number used on a machine, open `<OracleApplicationServer_Home>\install\portlist.ini` and read the value specified for “Application Server Control”.

The value of the Oracle Home for the remote server is the installation directory of the Oracle Application Server middle tier software.

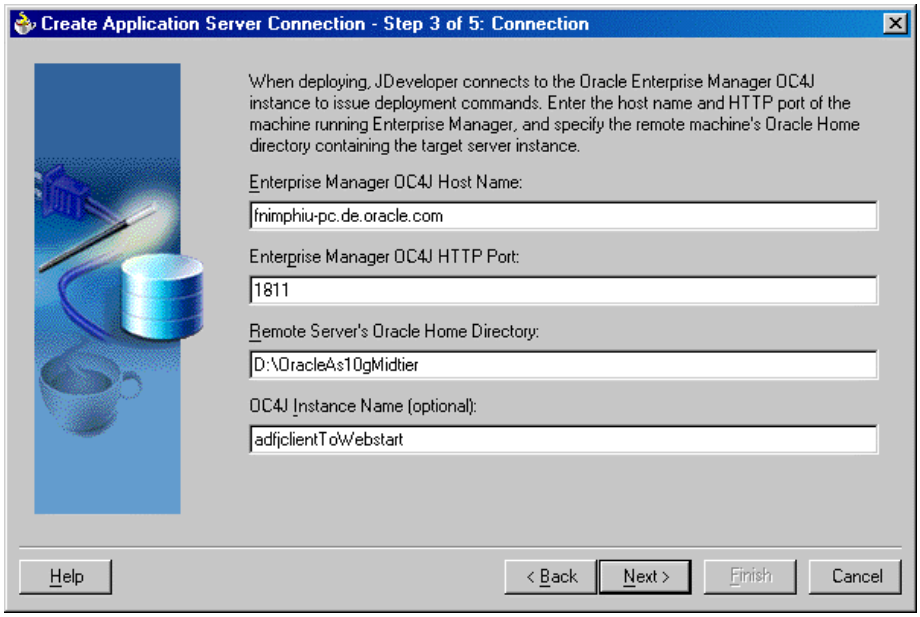

Figure 18: Provide Oracle Application Server Control 10*g* and OC4J instance information

The name of the OC4J instance that this named connection deploys to can be set as the "OC4J Instance Name" property. If not using a specific OC4J instance for deploying ADF JClient, the OC4J instance field can be left blank.

The forth wizard panel is similar to the one shown in the standalone deployment and needs the host name, port number, and connect information for the RMI connection.

The RMI port number used by an OC4J instance in Oracle Application Server can be looked up using Oracle Application Server Control 10*g*. The username and password for the RMI connection are defaulted to admin/ welcome[12].

The OC4J instance must be started for Oracle Application Server Control 10*g* to be able to show the RMI port used. RMI ports will change with each OC4J instance creation, even if a previously existing OC4J instance name is used.

[12] You should change the RMI password for production environments

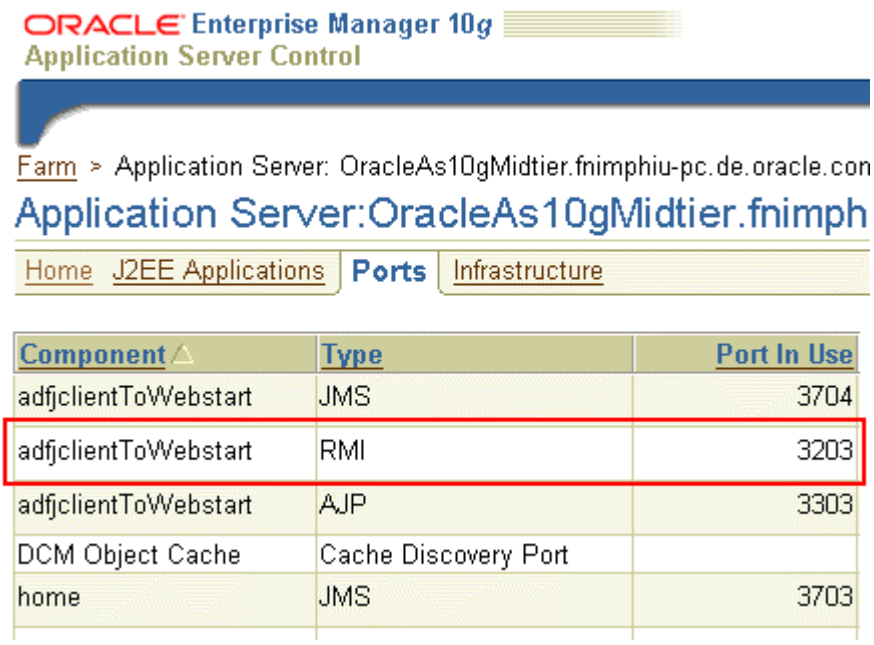

Figure 19: RMI Port usage shown in Oracle Application Server Control 10*g*

Finally, test the connection. If the test fails, verify the correctness of the account information and the server name provided.

The deployment of ADF JClient applications to Oracle Application Server 10g is the same as when deploying to standalone OC4J. Select the `bc4jlibs.ear` file entry in the JDeveloper Application Navigator and select **Deploy to → *<Oracle Application Server Connection Name>*** from the right mouse menu.

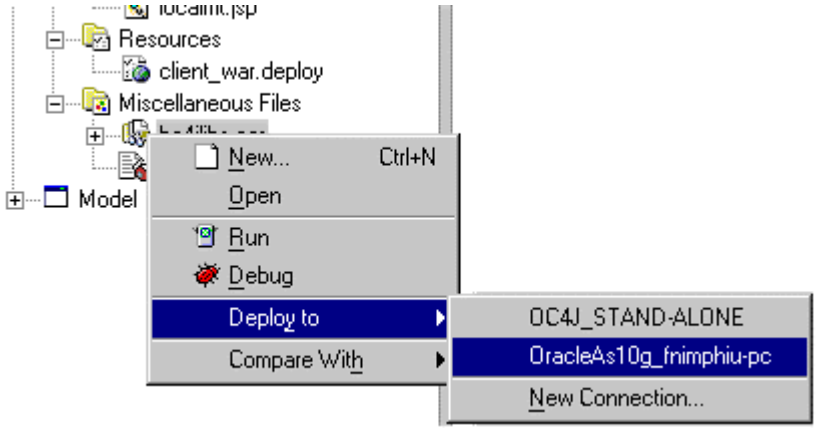

Figure 20: Deploying the bc4jlibs.ear file to the Oracle Application Server

To deploy the ADF JClient application, select the `client_war.deploy` profile in the Application Navigator and choose **Deploy to → *<Oracle Application Server Connection Name>*** from the right mouse menu.

Please refer to the section "Deploying the ADF JClient application to OC4J" on how to customize the web application deployment.

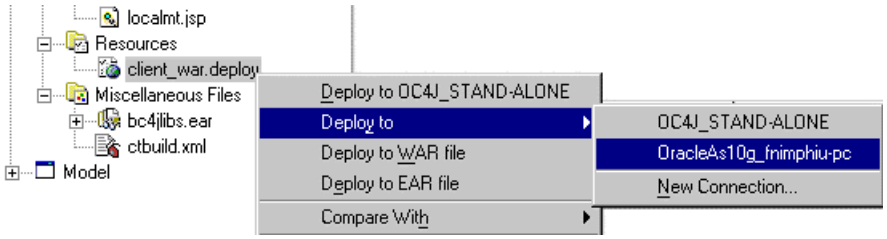

Figure 21: Deploying the ADF JClient application

To start the ADF JClient application, issue the following URL in a browser

```
http://server:port/<context root>/local.jsp
```

For example:

```
http://fnimphiu-pc.de.oracle.com:7777/adfjclientToWebstart
/local.jsp
```
[13]

The port number for the middle tier server can be looked up in *<OracleApplicationServer_Home>*`\install\portlist.ini`.

Local Deployment to Oracle Application Server 10*g* using OEM

Though you can use Oracle JDeveloper 10*g* to deploy ADF JClient applications to Oracle Application Server 10*g*, administrators will prefer to use Oracle Application Server Control 10*g*. This section explains how to deploy ADF JClient for Java Web Start to Oracle Application Server 10g in a local deployment.

The following two files need to be deployed to Oracle Application Server 10*g* to run ADF JClient:

- `bc4jlibs.ear`
- `client_war.war`

To create the `client_war.war` file, select the `client_war.deploy` entry in the JClient project using the right mouse button and choose the **Deploy to WAR File** menu option. This creates the `client_war.war` file in the "deploy" directory of the JClient project.

Open Oracle Application Server Control 10*g* in a web browser and select the link for the middle tier instance and then the OC4J instance the application should be deployed to. To deploy the `bc4jlibs.ear` file, click the **Deploy EAR File**button and browse the file system for the `bc4jlibs.ear` file in the

[13] It's a coincident that the name of the Web Application root context and the name of the OC4J instance created are the same. The application root context must not be the same as the OC4J instance.

<JDeveloper_Home>\Bc4j\jlibs directory. Provide a name for the deployment and finish the wizard, accepting the default settings on all other screens.

To deploy the client_war.war file, click the **Deploy WAR File** button and browse the file system for the client_war.war file located in the *<JClient_Project>*\deploy directory. Specify the J2EE context root, the virtual path name used when calling the ADF JClient application, and a name for the deployment. Finish the dialog to deploy the application.

The application is accessible from a web browser, using the following URL:

```
http://<server>:<port>/<context root>/local.jsp
```

Using static JNLP files

By running the Java Web Start wizard, static JNLP files can be created instead of the JSP files to start the ADF JClient application in local mode. Because the JNLP file is static, when deploying the ADF JClient Java Web Start application to OC4J or Oracle Application Server that run on a different server machine than JDeveloper, some information it contains needs to be changed:

1. The local.jnlp file contains a reference to the IP address and the port number of the embedded OC4J server in JDeveloper. This needs to be changed to the IP address or server name and the port number used by standalone OC4J or application server
2. The localmt.jnlp file also contains references to the IP address and the port used by the embedded OC4J server. Change this accordingly.
3. The application's context root name contained in the local.jnlp and the localmt.jnlp file is determined by the context root information of the JClient project.

   To change the context root name before creating the static JNLP file, select the JClient project and then the **Project properties** option from the right mouse menu. Select the **J2EE** entry and change the value for the **J2EE Web Context Root**. The root context name can also be changed manually in the JNLP files.
   In either case, deploying the Java Web Start application to Oracle Application Server using Oracle Application Server Control 10*g*, the same root context name needs to be provided in the deployment dialog.

Using static JNLP files will work with any version of Internet Explorer, while the JSP based JNLP file creation requires Internet Explorer 6.0.

THREE-TIER DEPLOYMENT

The ADF Business Components model can also be deployed as Enterprise Java Session Beans (EJB), allowing the ADF JClient application to run in a three-tier architecture.

In contrast to the local deployment explained earlier, the three-tier deployment only loads the ADF JClient-specific application files to the user desktop, thus reducing the overall download size.

To switch from one deployment type to the other, any business code added to ADF Business Components must be platform independent. This means that business functions should be accessed through the application module and not directly.

When running Business Components deployed as EJB, the decicion if using immediate validation or batch mode has an influence to performance.

When building J2EE applications with ADF JClient and ADF Business Components, there is no need to rewrite an application or parts of it to switch from one deployment form to the other.

Local deployment:

- Can run standalone
- Real client-server mode
- All application logic and validation performed on the client, thus less network traffic

Three-tier deployment:

- Business logic can be shared with other views like JSP, UIX and JSF
- Model patches are easier to implement
- Oracle Application Server Control 10*g* support
- Minimal download size
- Data can be prefiltered on the server

Prerequisites

Running ADF Business Components as EJBs on the server requires the ADF framework runtime libraries to be installed. Oracle JDeveloper 10*g* assists in deploying the ADF runtime libraries by providing an ADF Runtime Installer, accessible from the "Tools" menu.

The ADF framework runtime libraries can be installed not only to the Oracle Application Server, but also to Tomcat, JBoss and WebLogic.

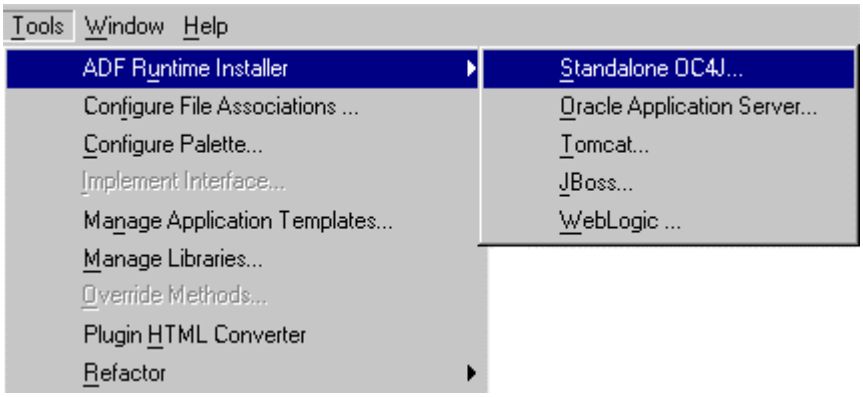

Figure 22: "ADF Runtime Installer" in Oracle JDeveloper 10*g*

Configuring ADF Business Components for EJB deployment

To deploy ADF Business Components models as EJB session beans, an EJB deployment profile needs to be created for the Application Module. Business

Components Application Modules can have multiple deployment profiles, which means that creating a deployment profile for EJB doesn't remove an existing configuration for local deployment.

Deployment profiles can also be created manually using the "Configurations" option of the application module context menu.

The deployment profile wizard is accessed through the **Business Components Deployment…** option in the right-mouse context menu on the Application Module node.

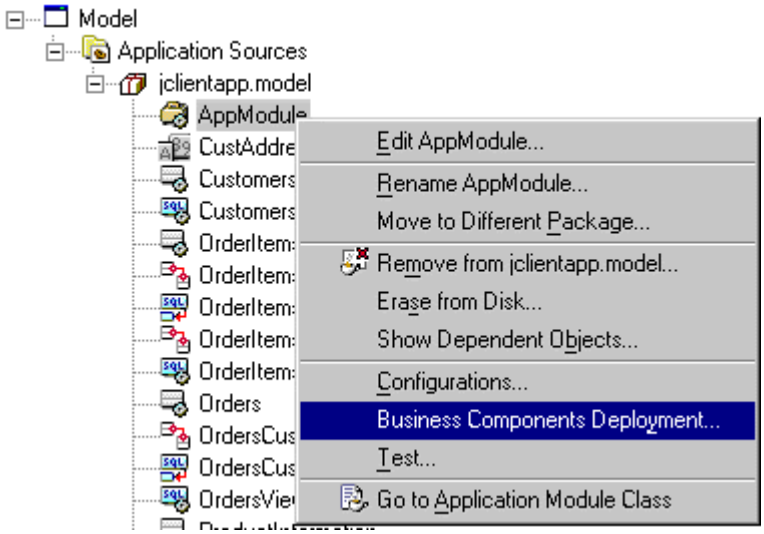

Figure 23: Creating a Business Component deployment profile from the Application Navigator

The default configuration is to deploy Business Components in a local deployment using "Simple Archive Files".

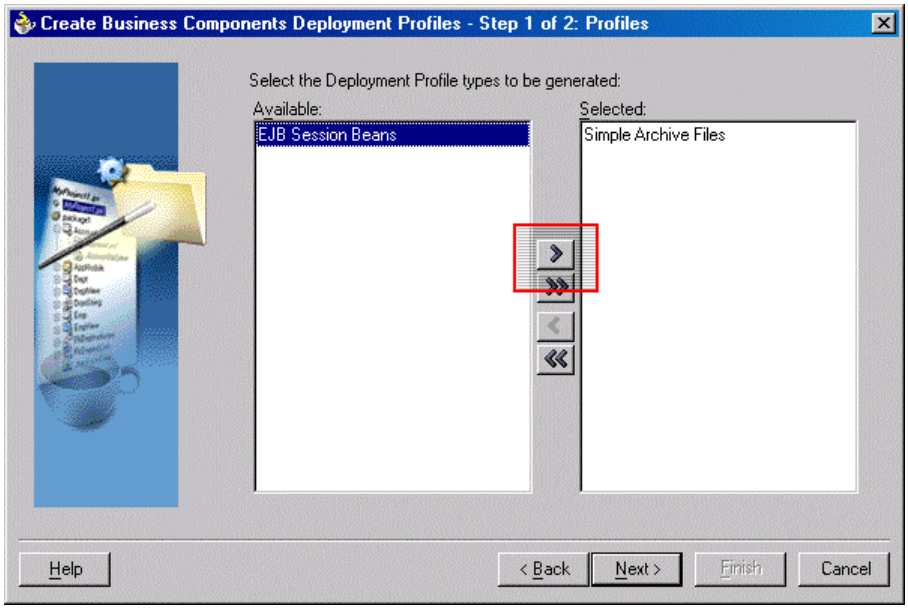

Figure 24: Selecting EJB as the profile to create

To create a profile for deploying the application module as an EJB session bean, select the EJB entry in the **Available** list and press the **>** button. The number of wizard steps varies with the selection.

The **Deploy to Connection** defines the runtime target when testing the EJB deployment using JDeveloper's integrated Business Components tester.

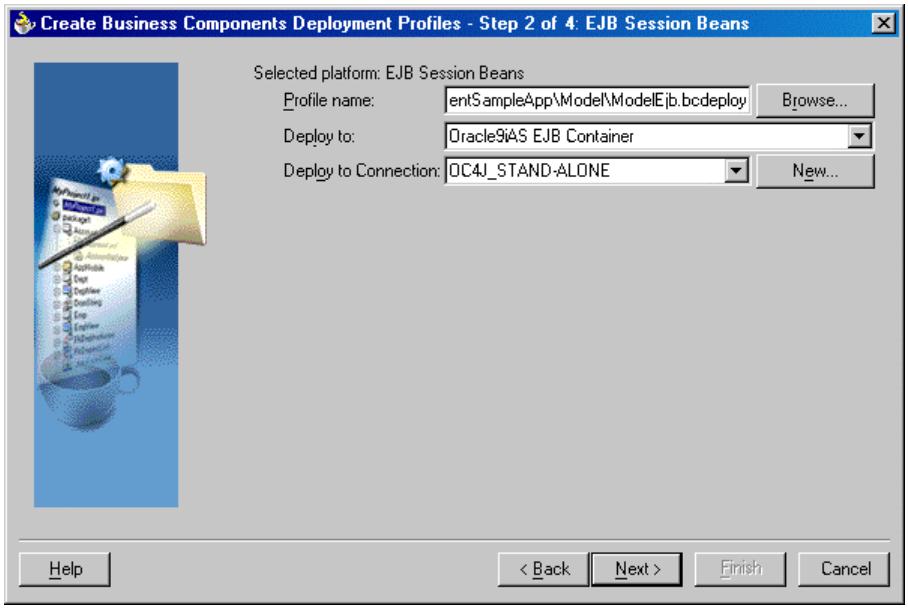

Figure 25: Selecting deployment type and connection

Select the Application Module to deploy as EJB and accept the default settings. The EJB Name will be created as *"<Application Module Name>*Bean".

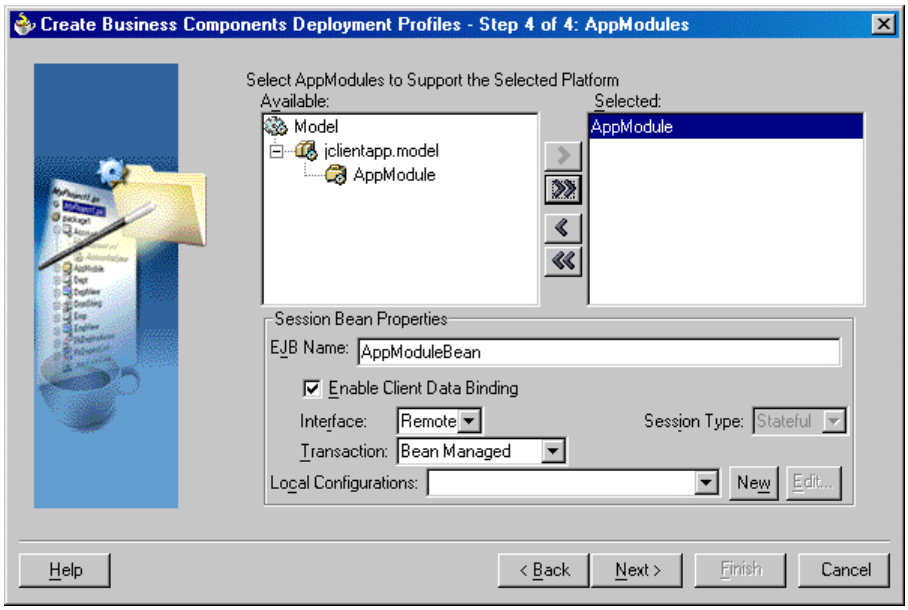

Figure 26: Selecting the Application Module for EJB deployment

The EJB deployment profile is created when finishing the deployment wizard and can be accessed from the Resources node of the Business Components, using the Application Navigator (Ctrl+ Shift+ A) in JDeveloper.

To deploy the Business Component model to the J2EE container, select the ***<Model Project Name>*EJB<n>.deploy** entry in the Resources node and choose a deployment option from the right-mouse button context menu.

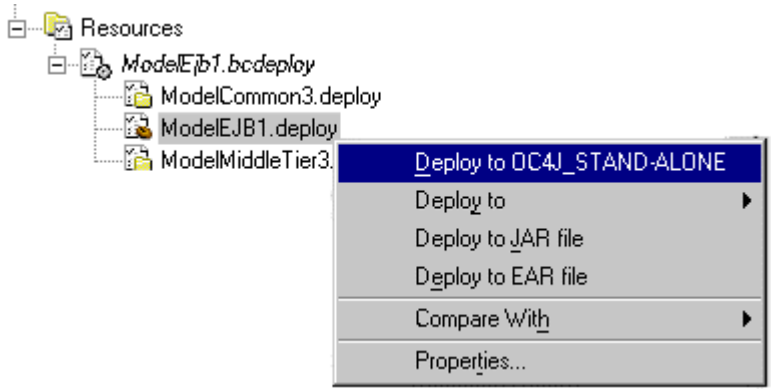

Figure 27: deploying Business Components as EJB

The deployment profile can be customized (e.g. to specify a different J2EE context root name than the default) through the **Properties** option in the context menu.

Testing ADF Business Components deployed as EJB

The Business Components EJB deployment can be tested with the integrated tester in Oracle JDeveloper 10*g*.

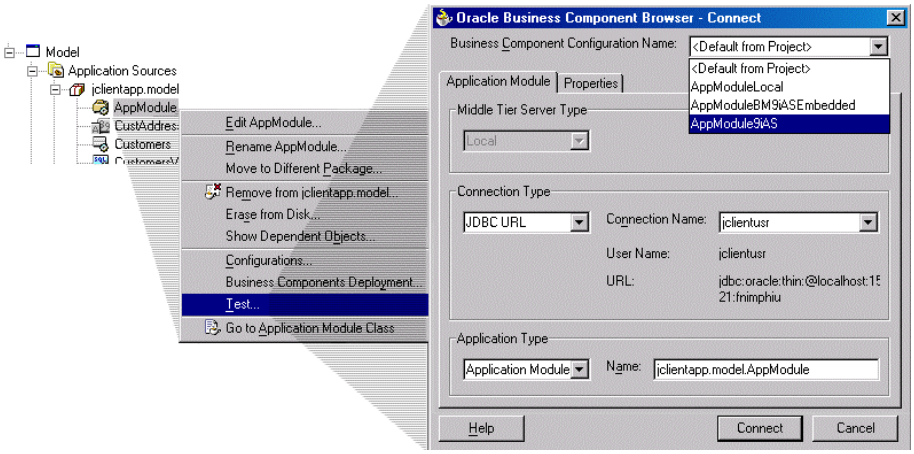

Figure 28: testing the Business Components EJB deployment.

To test the deployment, select the Business Components application module, "AppModule" in the image above, and click to the **Test** option in the right-mouse context menu.
In the Business Component Tester select ***<Model Project Name>*9iAS** as the Business Components Configuration name, "AppModule9iAS" in the example shown in the image above. Note that the **Connection Name** changes to a

reference in the `data-sources.xml` file deployed to the J2EE container. Click the **Connect** button to run the tester.

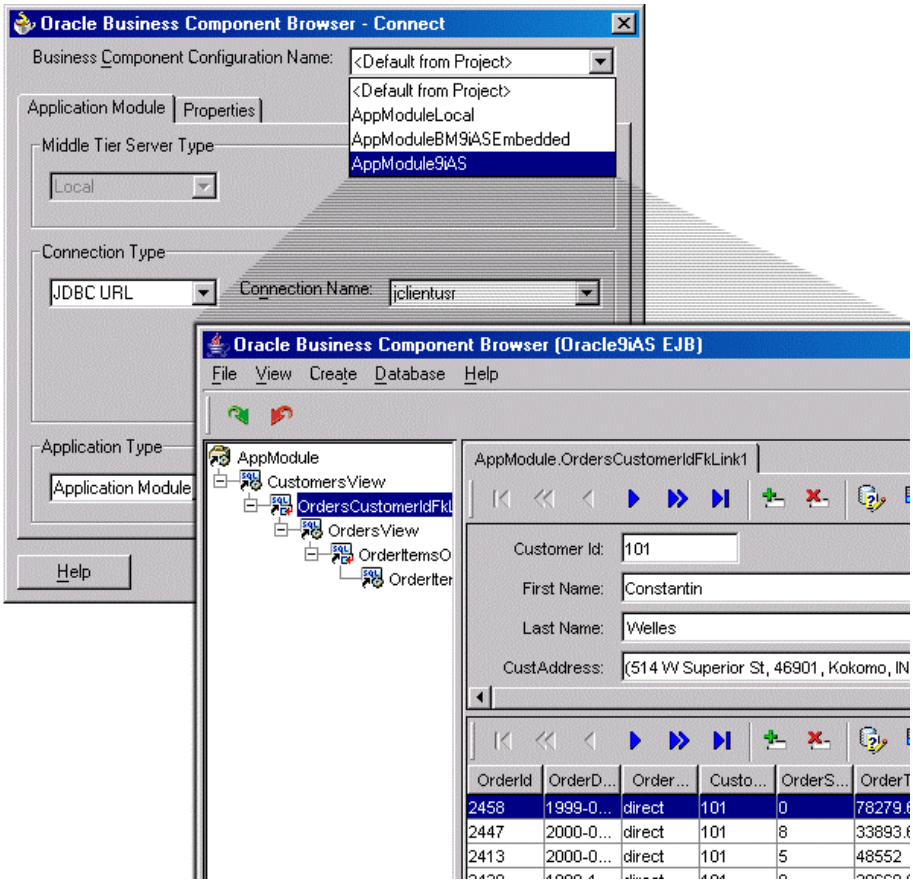

Figure 29: Business Component tester

☑ **Note:** To prove that the tester runs the Business Components EJB deployment, shutdown the J2EE container and perform any operation in the tester.

To shutdown the standalone OC4J server, type the following command on a command line in the OC4J's `j2ee/home` directory:

```
java –jar admin.jar ormi://<server>:23791 admin <password> -
shutdown
```

☑ **Note:** If the default ORMI port 23791 doesn't work, check the `rmi.xml` configuration file in the `OC4J j2ee\home\config` directory for the port used.

Testing ADF JClient with Business Components as EJB

So far, an Enterprise JavaBean deployment profile for Business Components was created, deployed to OC4J standalone, and tested using the Business Component tester tool.

As an optional test before deploying the application, it can be verified that the ADF JClient application itself works with the Business Components EJB deployment.

To test the ADF JClient application, edit the `DataBindings.cpx` file in the JClient project and configure it to use the EJB deployed Business Components model.

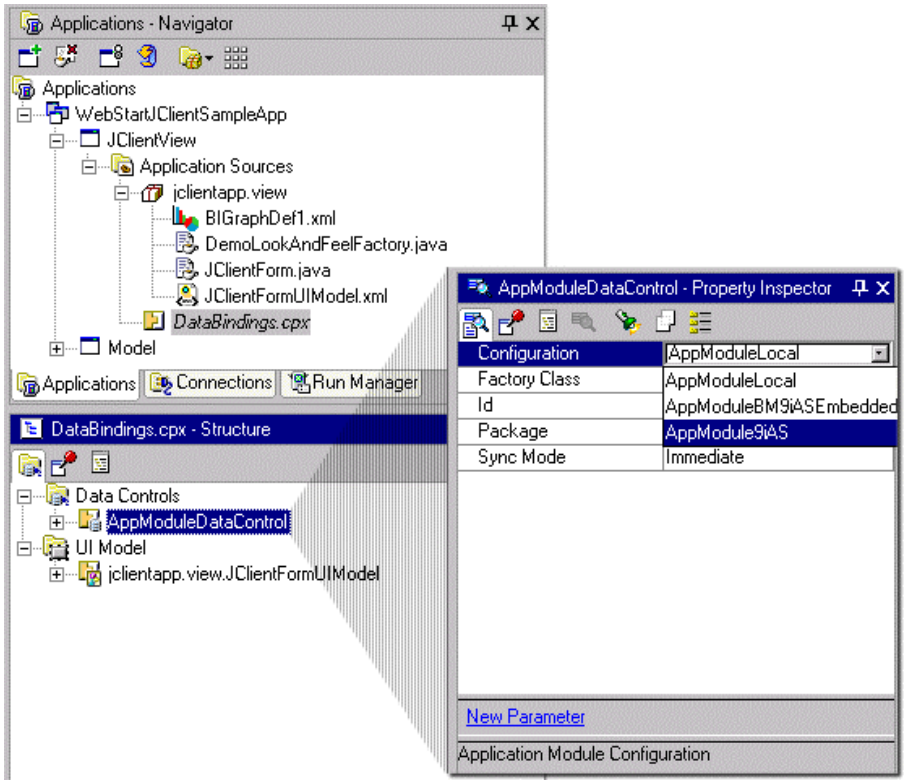

Figure 30: Switching the Business Components configuration used by JClient

In the ADF JClient project, select the **DataBindings.cpx** node and open the Structure window (Ctrl+ Shift+ S).

Select the Application Module Data Control entry, "AppModuleDataControl" in the image above, and open the Property Inspector (Ctrl+ Shift+ I). Change the **Configuration** property value to *<ApplicationModuleName>*`9iAS`[14].

Save the JClient project and run it to test or debug the application running with ADF Business Components deployed as EJB.

Application deployment to stand-alone OC4J

So far, the Business Component model was configured and deployed as an EJB and the ADF JClient application, run from the Oracle JDeveloper IDE, and was

[14] The may be more than one entry for <Application Module Name>9iAS showing in the list of values if multiple EJB configurations exist.

tested with this model. Next step is to also deploy the ADF JClient application to the standalone OC4J server.

In case the previous section was not read, please make sure the `DataBindings.cpx` file in the JClient project is configured to use the EJB deployment of the Business Components model. It is important to save the ADF JClient project after updating the `DataBindings.cpx` file.

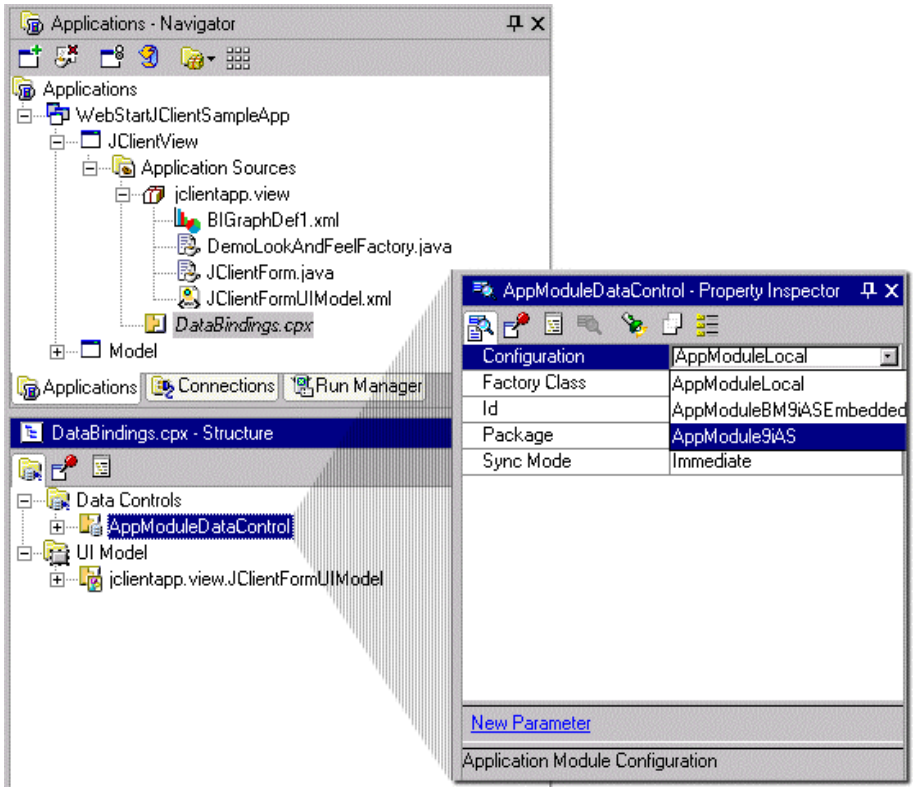

Figure 31: Configuring the DataBindings.cpx file to use the "<Application Module Name>9iAS" configuration.

To start the Java Web Start wizard, select the ADF JClient project in the Application Navigator (Ctrl+Shoft+A) and choose **New** from the right-mouse button context menu to open the New Gallery. Select the **Swing/JClient for ADF** category in the Client Tier and select **Java Web Start Files (JNLP) for JClient** in the list of available items. Click **Ok**.

Follow the wizard and accept the default setting for the "Data Module Definition". In the image above the data module definition name is "AppModuleDataControl".

Select **Generate Application Description** in step 2 of the wizard and provide information about the ADF JClient application start class, the title to show on the Java Web Start splash screen, the application description, and the vendor information. All the descriptive information is added to the JNLP file.

Finally, choose whether to start the Java Web Start application with a JSP file that dynamically creates the JNLP file or with a generated, static JNLP file[15].

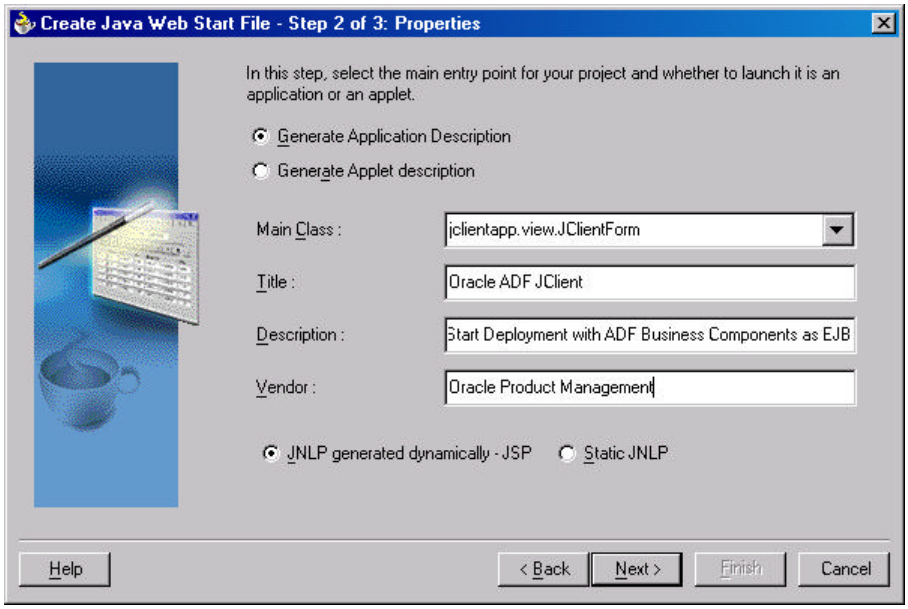

Figure 32: Defining the Java Web Start deployment properties

If the ADF JClient application contains graphs, add the Oracle Business Intelligence Graph bean libraries by checking the checkbox in the 3rd wizard panel.

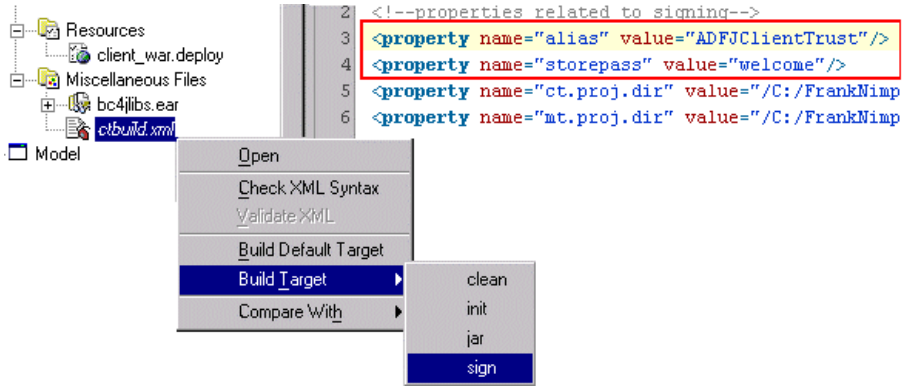

Figure 33: Running the ANT build script to create the client deployment profile.

To create a signed version of the client library files, open the `ctbuild.xml` ANT script and edit the `alias` name property value and the `storepass` property value to match the key created earlier.

[15] In most cases using a JSP file is preferred. However, in Microsoft IE releases below 6.0, there exists a known issue that prevents the JSP file from working without manual changes in the client registry. In this case, either use static JNLP files or Netscape.

Select the **ctbuild.xml** node in the Application Navigator and choose **Build Target|sign** from the right-mouse menu.

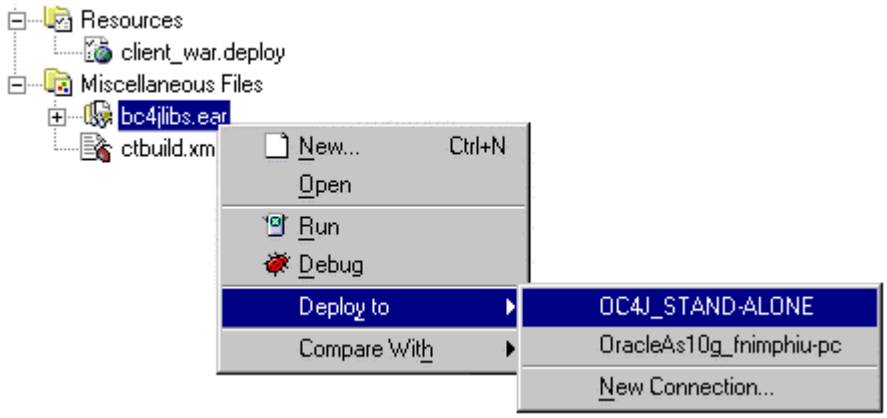

Figure 34: Deploying the bc4jlibs.ear file to OC4J

Before deploying the ADF JClient application to OC4J, deploy the Business Components libraries in the `bc4jlibs.ear` file.

Optionally, change the name for the application and the web context toot, by clicking on the **Properties** option in the right-mouse button menu of the `client_war.deploy` entry in the ADF JClient project.

The application name is what shows as a directory information in the target J2EE container, while the web context root defines the virtual path used to access the Web application.

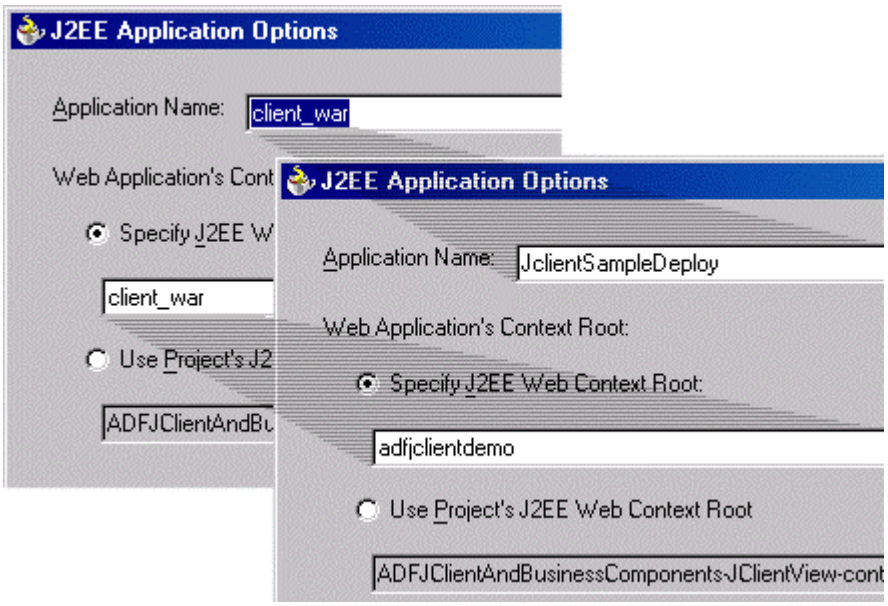

Figure 35: Optionally change the default name for the Application Name and the Web Context Root

To create a Web Archive file (WAR) for the client libraries, select the **client_war.deploy** node in the JClient project and select **Deploy to WAR file** from the right-mouse button menu.

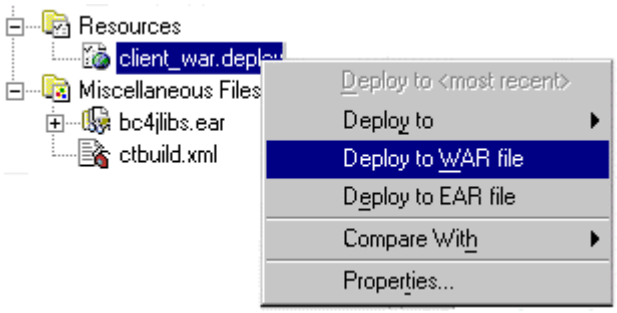

Figure 36: Creating a client deployment WAR file

Add the created `client_war.war` file to the project by using **File|Add** to the JClient project or clicking the button. The WAR file's location is in the `deploy` directory of the JClient project on the file system. The WAR file does not need to be added to the project's source path.

Select the `client_war.war` file entry in the Application Navigator and use the right-mouse button to open the context menu. Select the **Deploy to…** option in the context menu to deploy this file to the OC4J instance.

After a successful deployment, the JDeveloper log message window will show the access URL for your application, which looks similar to the following when applying the changes suggested in figure 35.

```
"…
Use the following context root(s) to test your web
application(s):

http://<your server name>:8888/adfjclientdemo
…"
```

The JSP application start file generated by the Java Web Start wizard is `oc4j.jsp`, so that the application request URL becomes

```
http://<your server name>:8888/adfjclientdemo/oc4j.jsp
```

Application deployment to Oracle Application Server 10*g*

Deploying ADF JClient with a Business Components EJB model to Oracle Application Server is similar to the deployment to the standalone OC4J server.

As with OC4J standalone, if it's the first time that an ADF application is deployed to Oracle Application Server 10*g*, the ADF runtime libraries need to be installed using the ADF Runtime Installer provided in Oracle JDeveloper 10*g*.

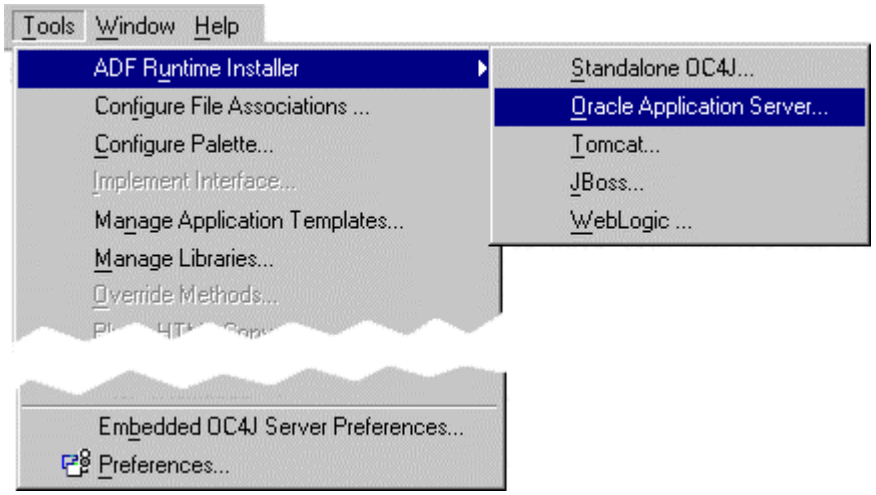

Figure 37: Installing ADF Runtime Libraries to Oracle Application Server 10*g*

During the course of this whitepaper, a deployment configuration for deploying the ADF Business Components model as an EJB to OC4J standalone has been created. In order to deploy to Oracle Application Server this configuration can be modified or an additional configuration can be created. For this paper, the second option is chosen, allowing to later run and test both deployments from same the Business Components project.

Creating a Business Components EJB deployment profile for Oracle Application Server includes the same configuration steps as explained in the section titled "Configuring ADF Business Components for EJB deployment".

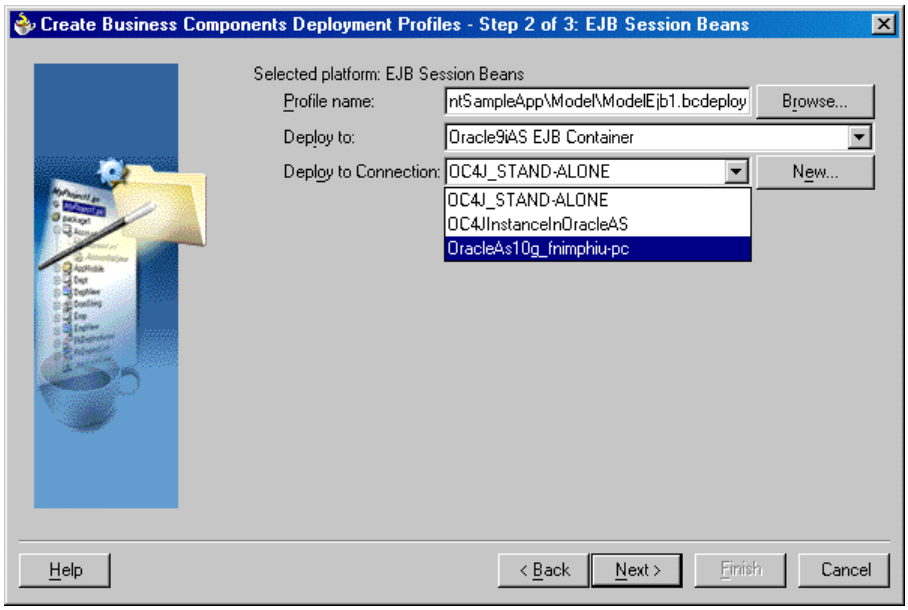

Figure 38: Creating a Business Components deployment profile for Oracle Application Server 10*g*.

The only difference is that in step 2 of the profile wizard the named connection for the application server 10g[16], `OracleAs10g_fnimphiu-pc` in the example shown above, is selected[17].

It is important that the application server connection contains the correct RMI port used by the OC4J instance in the Oracle Application Server. The port number can be looked up using Oracle Application Server Control 10*g* (*Figure 19*).

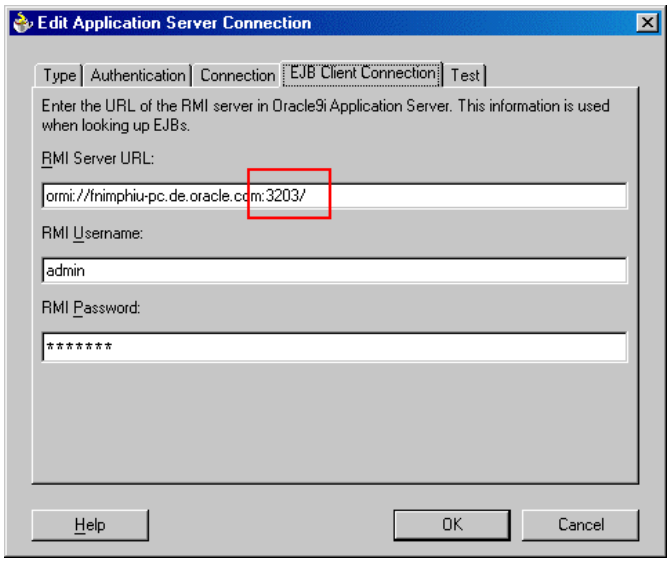

Figure 39: Server connection for OC4J instance in Oracle Application Server

Finish the wizard to create a new deployment configuration for the application module.

A new deployment profile is created under the **Resources** node of the Business Components project.

Deploy the ***<ApplicationModuleName>*EJB<n>.deploy**[18] entry to Oracle Application Server using the right-mouse button menu and selecting the named application server connection.

Testing the Business Components deployment

To test the ADF Business Components model deployed as EJB to Oracle Application Server, select the application module in the Business Components model using the right-mouse button and choose **Test** from the context menu.

---

[16] If there doesn't yet exist a connection for Oracle Application Server in JDeveloper, please create one first as described in the section titled "Local Deployment to Oracle Application Server 10*g*".

[17] In the section titled "Local Deployment to Oracle Application Server 10g", it is explained how to create a named connection form Oracle Application Server 10g. In the following the created connection "OracleAs10g_fnimphiu-pc" will be used as an example.

[18] If more than one deployment configuration was created, the the different ".deploy" files are distinguished by a sequential numbering (<n>)

In the test dialog, select the Business Components profile created for the application server deployment and make sure that the **Oracle9AS Server<n>**[19] selection box shows the named connection of Oracle Application Server 10g.

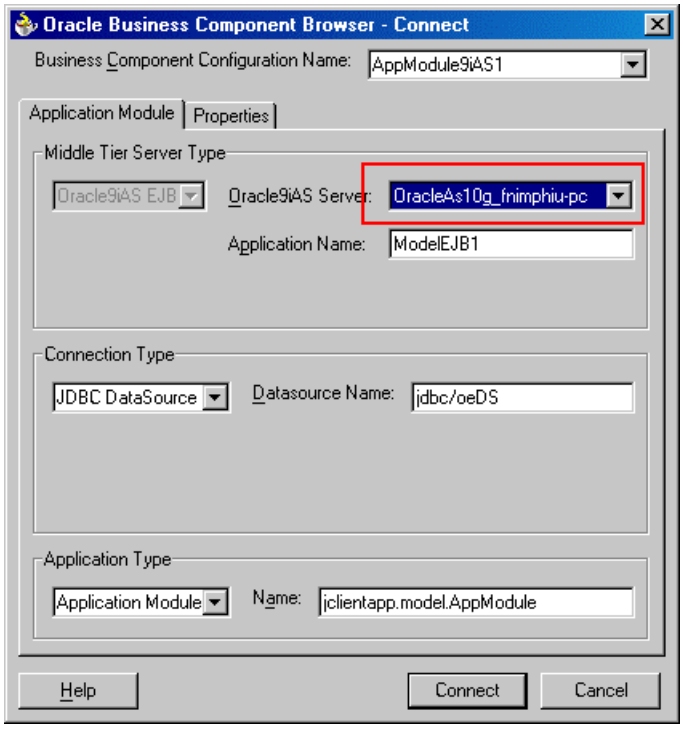

Figure 40: Selecting the Application Server 10g connection in the Business Component tester to test the model gainst the deployed EJB sources

Click the **Connect** button to start the tester.

☑ **Note**: If the **Oracle9AS Server** selection box doesn't show the connection name configured for the OC4J instance in Oracle Application Server 10g, do the following

1. Select the application module entry in the Business Components project.
2. Choose **Configurations** from the right-mouse menu.
3. Select the deployment configuration name created to be used with Oracle Application Server deployments.
4. Click the **Edit** button.
5. Set the **Oracle9iAS Server** selection box value to the name of the OC4J instance in Oracle Application Server.
6. Click **OK**.

The section titled "Testing ADF JClient with Business Components as EJB" explains how to run and test an ADF JClient application with a Business Components model deployed as EJB in Oracle JDeveloper.

[19] If more that one EJB configuration exist for a Business Components model, a number suffix is used to determine the different configurations.

☑ **Note:** For Business Components deployed to application server, make sure that the checkbox **Deploy Password** is checked for the ORMI password. Otherwise the JClient application will not start.

Deploying the ADF JClient application

Creating the Java Web Start deployment descriptor and deploying ADF JClient to the Oracle Application Server 10g is the same as explained in the section titled "Deploying ADF JClient to standalone OC4J".

Make sure that the `DataBindings.cpx` file in the ADF JClient project is updated to use the correct Business Components configuration and that the project is saved before running the Java Web Start wizard.

Deployment to Oracle Application Server 10g using OEM

Though Oracle JDeveloper 10*g* can be used to deploy ADF JClient applications to Oracle Application Server 10*g*, administrators may prefer Oracle Application Server Control 10*g*.

This section explains the deployment of ADF JClient and ADF Business Components using Oracle Application Server Control 10*g*.

The following assumptions are made:

1. The ADF runtime libraries are installed in Oracle Application Server 10*g* using the ADF Runtime Installer in Oracle JDeveloper 10g. This is explained in the section titled "Three-tier deployment" *(page 24)*.
2. An EJB deployment configuration was created for the ADF Business Components application module, as described in the section titled "Three-tier deployment" *(page 24)*.
3. ADF JClient is configured to work with the Business Components EJB profile having set the `Configuration` property of the Data Control entry in the `DataBindings.cpx` file. This is explained in the section titled "section "Three-tier deployment" *(page 24)*.

The ADF Business Components project structure should look similar to what is shown in the image below:

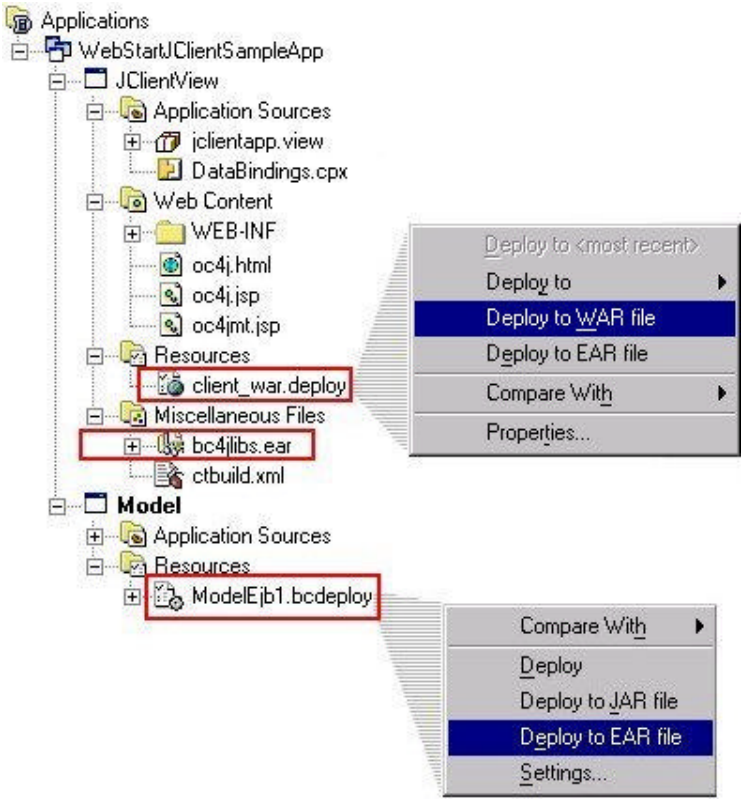

Figure 41: Deployment profiles

Create a Web Archive file (WAR) from the **client_war.deploy** deployment descriptor, using the right-mouse button menu option. The WAR file will be created in the "`Deploy` directory of the JClient project directory on the file system.

Similarly, create an Enterprise Archive file (EAR) from the **bcdeploy** deployment descriptor. The EAR file will be created in the `Deploy` directory of the ADF Business Components project directory on the file system.

Open Oracle Application Server Control 10*g* and select the Application Server middle tier link on the main page. In the application server web site of the middle-tier installation, select the name of the OC4J instance[20] that the application shall be deployed to. If the OC4J instance isn't started, start it now.

Deploy the different deployment files using the **Deploy EAR file** and **Deploy WAR file** buttons in the following order:

1. `bc4jlibs.ear`, located in the *<JDeveloper_Home>*`\Bc4j\jlibs` directory
2. Business Components EJB EAR file (`*.bcdeploy`)
3. `client_war.war`

[20] Best practice is to create a separate OC4J instance for experimenting with the ADF JClient deployment. This allows to undo any modifications without impacting other applications run on Oracle Application Server.

Figure 42: OC4J instance with ADF JClient / ADF Business Component deployment

To deploy the EAR files, click the **Deploy EAR file** button and select the EAR file on the file system. Provide a name for the deployment in the application server and in the following accept all default settings.

To deploy the WAR file, click the **Deploy WAR file** button. In the first deployment page, define a name for the application deployment and a URL mapping for the deployed application. The URL mapping defines the virtual path used to access the application from a web browser.

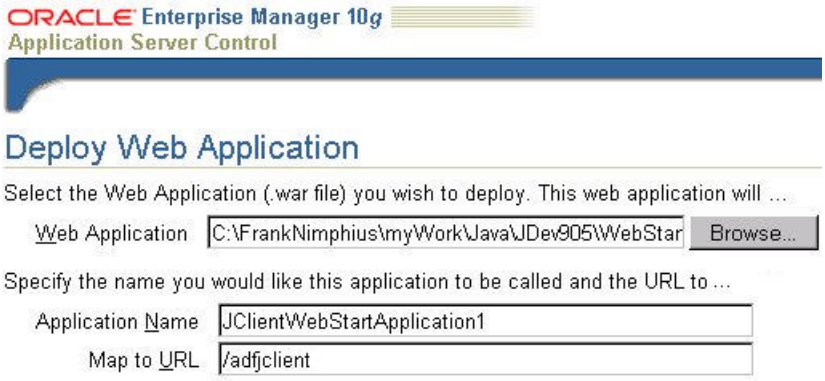

Figure 43: Deploying the client WAR file and defining the URL mapping

In the example shown in the image above, the ADF JClient application deployment uses a URL mapping of `/adfjclient` and thus can be accessed as follows:

```
http://server:port/adfjclient/oc4j.jsp
```

DEPLOYING ADF JCLIENT TO JBOSS

The JBoss application server originates from an open source project and has wide adoption within the J2EE community. The JBoss 3.x series by default now includes Tomcat 4.1.29 as a web engine.

This whitepaper does not focus on how to configure and run JBoss but instead assumes the reader is familiar with this application server[21].

The intention of this section is to explain the deployment of ADF JClient and ADF Business Components to an application server that is not Oracle.

As before, the application is supposed to run using Java Web Start as a client runtime environment.

Install ADF Runtime Libraries

Use the **ADF Runtime Installer** option in the Oracle JDeveloper **Tools** menu to install the ADF runtime libraries to the JBoss server.

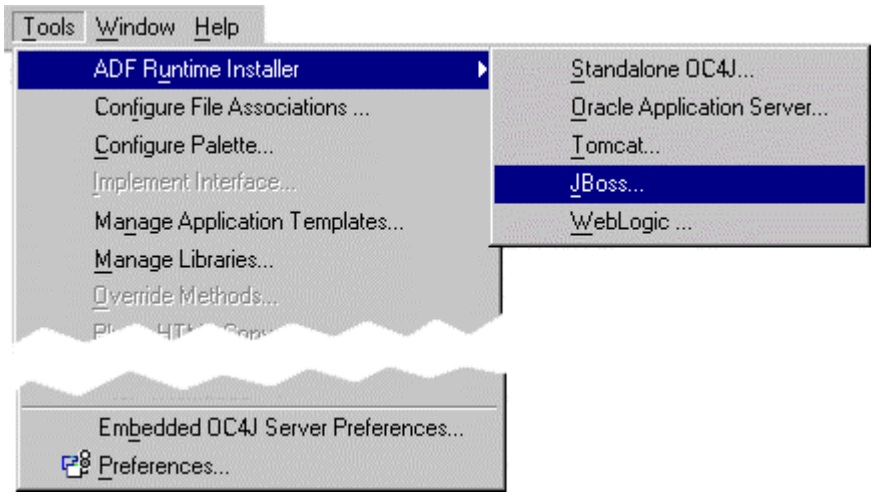

Figure 44: Install ADF Runtime libraries to JBoss

Restart JBoss after finishing the installer. Not restarting JBoss will cause subsequent EAR file deployments to fail.

Create a JBoss connection in Oracle JDeveloper

Before deploying the ADF JClient application to JBoss, an application server connection for JBoss needs to be created in Oracle JDeveloper 10*g*.

Open the Connection Navigator (Ctrl+Shift+O) in Oracle JDeveloper 10*g* and select the application server entry. Use the right-mouse button to open the context menu and select **New Application Server connection** to launch the connection wizard.

---

[21] Based on the author's experience when downloading and running JBoss 3.2.3, it's important to check the server.log file located in <JBoss-3.2.3>\ server\ default\ log for port conflicts after staring JBoss the first time. To test JBoss is working, you should be able to run http:/ / <server or localhost>:8080/ jmx-console, assuming the default port is used.

Provide information about the connection name and type, as well as the location of the server deploy directory of the JBoss installation, as shown in the image below.

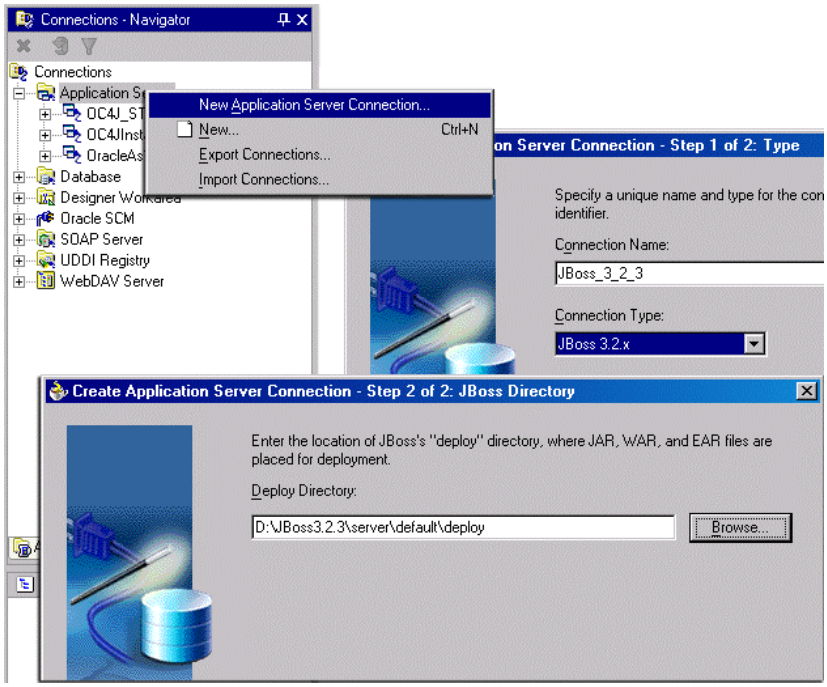

Figure 45: Creating a JBoss connection in JDeveloper

Finish the wizard to create the connection.

Local deployment

The local Java Web Start deployment is similar to the deployment described in the section titled "Local deployment to Java Web Start on OC4J".

It is assumed that a signed version of the `bc4jlibs.ear` file exists as described earlier in this whitepaper. If not, please read and follow the instructions in the section titled "Prerequisites " before continuing.

Creating a Business Component Deployment profile

For local deployment, a Business Components application module with a configuration profile for simple **Java Archive Files** is used. This configuration type is the default configuration available after creating the application module.

Creating the Java Web Start deployment file

Please see the section titled "Local deployment to Java Web Start on OC4J" for details about how to create a client deployment file for Oracle ADF JClient.

Local Deployment to JBoss

After creating the Java Web Start deployment profile, perform the following steps to deploy ADF JClient and the Business Components model to JBoss:

1. Select the `bc4jlibs.ear` file entry in the JClient project tree and choose **Deploy to | *<JBoss connection>*** from the right-mouse context menu
2. Select the **client_war.deploy** entry under the **Resources** node and deploy it to the JBoss connection.

   Note: To change or look up the context root directory used by the web application, select the **client_war.deploy** entry in JDeveloper and choose the **Properties** option from the right-mouse button menu.
3. Invoke the application on JBoss by pointing the web browser to the following URL:

   ```
   http://<server>:8080/<context root>/local.jsp
   ```

SUMMARY

This whitepaper showed various ways of deploying ADF JClient applications that use an ADF Business Components model and Java Web Start. Java Web Start is the preferred deployment environment for Oracle ADF JClient applications because it combines local deployment with the benefit of centralized software management. In addition, for intranet applications that do not require offline support, a three-tier deployment architecture can be configured to minimize the initial application download size.

TROUBLESHOOTING[22]

The following is a brief excerpt of troubleshooting tips that were found while writing this whitepaper. More information is available on the Java Web Start page on `http://java.sun.com`.

IE downloads the JNLP JSP file instead of launching Web Start

It's a known issue that Internet Explorer releases older before 6.0 try to download the Java ServerPages file generated to launch the Java Web Start Application. In this case, please generate static JNLP file to start ADF JClient on Java Web Start.

Initial application download hangs

Check whether the security dialog is hidden behind the Java Web Start progress screen. The application waits for the user to response to the security alert before continuing the download.

---

[22] More troubleshooting notes are available in the Java Web Start FAQ at http:/ / java.sun.com/ products/ javawebstart/ faq.htm

Error when downloading application library file

Make sure the user's desktop has enough free disk space for the application libraries downloaded to the client.

Cannot load resources over the network

This is likely to happen in a network environment, which has a firewall between the computer and the Internet or intranet and if Java Web Start has not been configured with the right proxy settings. Java Web Start can typically pick these settings up automatically by querying the system or the default browser.

Oc4jmt.jsp or localmt.jsp are not accessible

This problem experienced during local testing indicates that the server name referenced in the application URL is not known by the proxy defined in the browser settings. The first application access to `oc4j.jap` or `local.jsp` are successful, but the following, integrated, call to `oc4jmt.jso` or `localmt.jsp` fails. Use "localhost" or 127.0.0.1 as a server reference instead.

Cannot access Business Components deployed as EJB

If the error message states that the JNDI connection is refused, then most likely the ORMI port specified in the named Application Server connection used in JDeveloper is not correct.

When using an OC4J instance in Oracle Application Srever 10*g*, please use Oracle Application Server Control 10*g* to obtain the RMI port number used. Update the application server connection in JDeveloper with this information and recreate the application deployment files.

JBO-30003

The error message `JBO-30003: The application pool (jclientapp.model.<ModuleConfigurationName>) failed to checkout an application module` indicates that the database specified for the application is not available.

Browser shows JNLP file as plain text

This most likely occurss because your web server is not aware of the proper MIME type for JNLP files. Create the MIME type `application/x-java-jnlp-file` on your application server.

Deploying Oracle ADF JClient Applications with Java Web Start
May 2004
Author: Frank Nimphius, Aminur Rashid
Contributing Authors: Ralph Gordon

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com